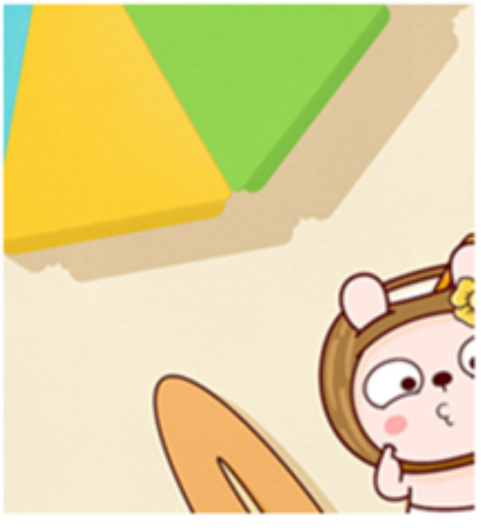
最新IT书籍PDF
最新IT书籍PDF
最新IT书籍PDF
最新IT书籍PDF



版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

**版权相关注意事项：**
**1、书籍版权归著者和出版社所有**
**2、本PDF来自于各个广泛的信息平台，经过整理而成**
**3、本PDF仅限用于非商业用途或者个人交流研究学习使用**
**4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负**
**5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF**
**6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅**
**7、请于下载PDF后24小时内研究使用并删掉本PDF**

版权所有，严禁传播，违者自负法律责任！

CETC 云计算技术与应用专业校企合作系列教材

非卖品，仅供非商业用途或交流学习使用

# Hadoop大数据平台构建与应用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

主编 米洪 陈彦

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

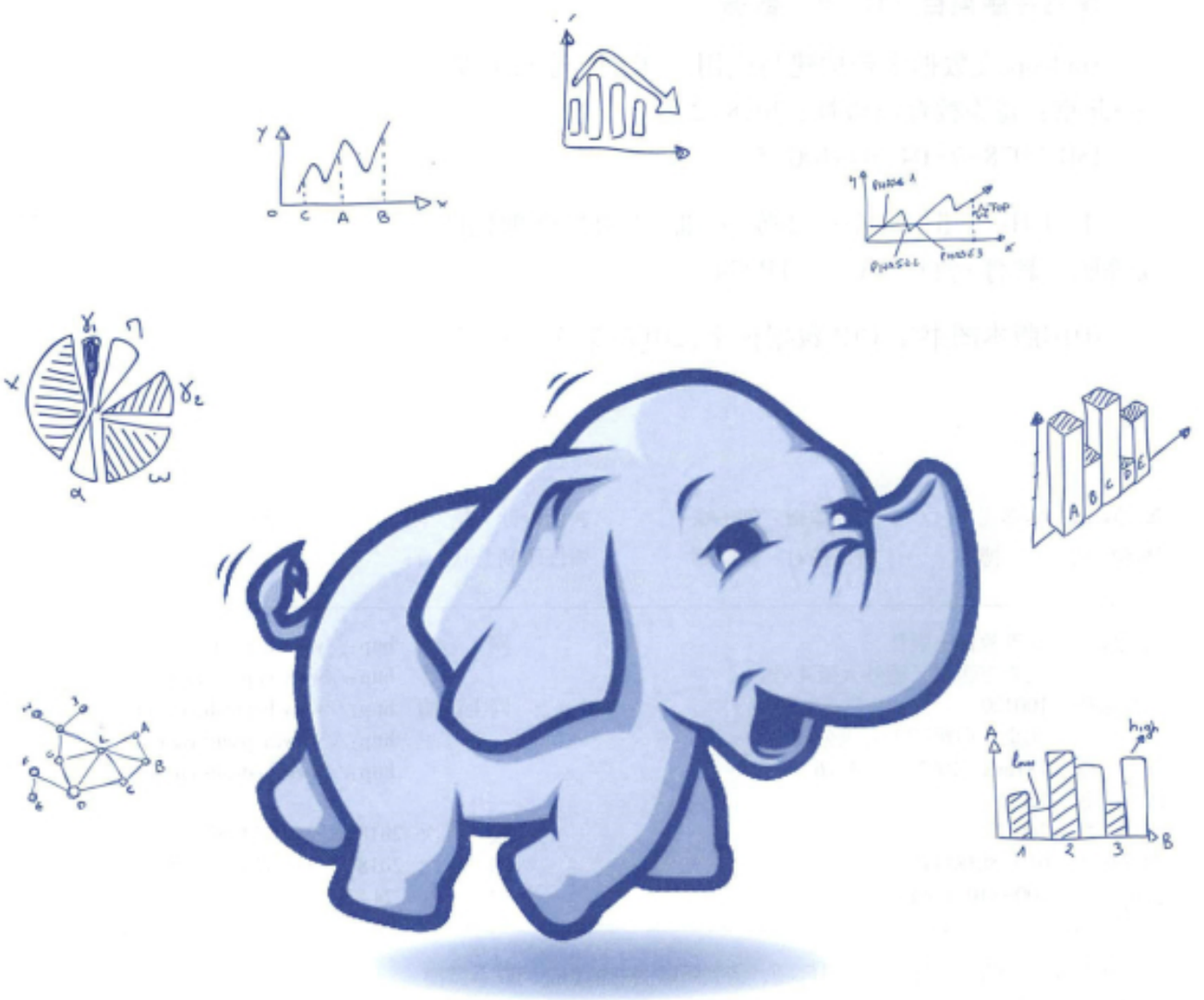
高等教育出版社

非卖品，仅供非商业用途或交流学习使用
严禁网络传播本PDF，违者责任自负！
版权所有，严禁传播，违者自负法律责任！

CETC 云计算技术与应用专业校企合作系列教材

# Hadoop大数据平台构建与应用

主 编 米 洪 陈 永



高等教育出版社·北京

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

内容提要

本书以“高职院校学情分析”系统为项目原型，以任务驱动为主线，基于 Hadoop 大数据平台，讲解大数据平台的搭建与运维、大数据的采集与存储、大数据的处理、数据的分析、数据的可视化等完整的大数据应用案例，全面详细地讲述 Hadoop、MapReduce、HDFS、Hive 和 Sqoop 等技术的相关知识，并详细讨论了 Mahout 大数据分析中的常用算法——K-means 聚类算法、Canopy 聚类算法、欧氏距离算法、余弦距离算法、皮尔逊相关系数算法、TF-IDF 加权算法等的使用及相关学情分析系统中算法的实现等。

本书具有较强的实用性和可操作性，语言精练，通俗易懂，操作步骤描述详尽，并配有大量操作图例。

本书可作为高等职业院校大数据应用专业、软件技术专业、云计算技术与应用专业的大数据分析与软件开发等相关课程的教材，也可作为从事大数据分析、云计算应用等系统开发与分析技术人员的参考用书。

**图书在版编目（CIP）数据**

Hadoop 大数据平台构建与应用 / 米洪，陈永主编. --北京：高等教育出版社，2018.2

ISBN 978-7-04-049100-5

Ⅰ.①H… Ⅱ.①米… ②陈… Ⅲ.①数据处理软件-高等职业教育-教材 Ⅳ.①TP274

中国版本图书馆 CIP 数据核字(2017)第 314265 号

| | | | |
|---|---|---|---|
| 策划编辑 吴鸣飞 | 责任编辑 刘子峰 | 封面设计 赵 阳 | 版式设计 童 丹 |
| 插图绘制 于 博 | 责任校对 窦丽娜 | 责任印制 耿 轩 | |

| | | | |
|---|---|---|---|
| 出版发行 | 高等教育出版社 | 网 址 | http://www.hep.edu.cn<br>http://www.hep.com.cn |
| 社 址 | 北京市西城区德外大街 4 号 | | |
| 邮政编码 | 100120 | 网上订购 | http://www.hepmall.com.cn<br>http://www.hepmall.com<br>http://www.hepmall.cn |
| 印 刷 | 北京市白帆印务有限公司 | | |
| 开 本 | 787mm×1092mm 1/16 | | |
| 印 张 | 12 | | |
| 字 数 | 260 千字 | 版 次 | 2018 年 2 月第 1 版 |
| 购书热线 | 010-58581118 | 印 次 | 2018 年 2 月第 1 次印刷 |
| 咨询电话 | 400-810-0598 | 定 价 | 24.80 元 |

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

版权所有 侵权必究

物 料 号 49100-00

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

**版权相关注意事项：**

**1、书籍版权归著者和出版社所有**

**2、本PDF来自于各个广泛的信息平台，经过整理而成**

**3、本PDF仅限用于非商业用途或者个人交流研究学习使用**

**4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负**

**5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF**

**6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅**

**7、请于下载PDF后24小时内研究使用并删掉本PDF**

版权所有，严禁传播，违者自负法律责任！

# 前　言

## 一、缘起

大数据技术在人们日常生活中已经得到了广泛的应用。信息技术与经济社会的交汇融合引发了数据迅猛增长，数据已成为国家基础性战略资源，大数据正日益对全球生产、流通、分配、消费活动以及经济运行机制、社会生活方式和国家治理能力产生重要影响。大数据正快速发展为对数量巨大、来源分散、格式多样的数据进行采集、存储和关联分析，

从中发现新知识、创造新价值、提升新能力，成为新一代信息技术和服务业态。

Apache Hadoop 是开源的大数据平台，基于普通服务器或者云平台构建分布式的大数据存储和基于 MapReduce 编程模型进行大数据处理。2017 年 3 月发布的 Apache Hadoop 2.8，表明 Hadoop 经过了 11 年的发展，已经建立了成熟的生态圈，并成为大数据行业的重要技术标准之一。

Apache Hadoop 项目包括以下 4 个主要模块。

① Hadoop Common：Hadoop 的通用工具集。

② Hadoop Distributed File System（HDFS™）：分布式文件系统。

③ Hadoop YARN：任务调度和集群资源管理框架。

④ Hadoop MapReduce：基于 YARN 的并行处理编程模型。

此外与 Hadoop 相互关联的开源大数据项目还有 Ambari、HBase、Hive、Mahout、Pig、Spark 等。

## 二、特点

本书以构建和开发“高职院校学生学情分析”系统为项目原型，以任务驱动为主线，指导学生基于 Hadoop 大数据平台，完成平台搭建与运维、大数据采集与存储、大数据处理、数据分析及数据可视化等完整的大数据应用案例。全书教学内容设计如下：

1. 大数据平台构建与运维

通过构建数据存储 HDFS、数据处理引擎 MapReduce、数据获取 Sqoop、数据处理 Hive、数据挖掘分析 Mahout 及 Spark 等系列服务与 Hadoop 平台运维，为大数据分析与应用奠定基础。

2. 招聘信息和学生成绩数据采集与存储

以典型招聘网站发布的岗位信息、学生在校各个管理平台成绩数据和近几年的学生就业统计数据为数据源，通过命令方式、开发程序、使用 Sqoop 等方式将数据分别存储到 HDFS 和 Hive 中为大数据分析做好充分准备。

3. 招聘岗位数据处理

通过 Hadoop 平台服务以及采集处理的岗位招聘数据，对数据进行清洗、过滤和查询分析，完成云计算、大数据岗位数据的处理。

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



4. 岗位和技能数据分析

通过 Hadoop 平台服务，使用 Mahout 命令进行岗位技能点聚类分析，并结合学生技能点数据的分析，给出相关工作岗位的推荐。

5. 数据可视化

通过对岗位对应技能点数据和学生实际掌握技能点数据的分析，为更好地展示数据分析结果，通过使用 Excel、Echarts、D3.js 等大数据可视化工具实现相关数据的可视化。

6. 学习资源推荐

根据岗位对应技能点数据和学生实际掌握技能点数据的分析，以及可视化展示结果，

结合互联网学习资源和相应就业岗位推荐数据，给出个性化的学习资源推荐。

## 三、使用

1. 教学内容课时安排

本书建议授课 64 学时，教学单元与课时安排见表 1。

表 1　教学单元与课时安排

| 序　号 | 项 目 名 称 | 学 时 安 排 |
|---|---|---|
| 1 | 认识大数据，实现学情分析系统设计与环境搭建 | 8 |
| 2 | Hadoop 大数据平台的构建 | 10 |
| 3 | 就业岗位数据的采集和存储 | 12 |
| 4 | 岗位数据的处理 | 12 |
| 5 | 岗位和技能数据的分析 | 12 |
| 6 | 数据的可视化 | 10 |
| 课时总计 | | 64 |

2. 课程教学资源一览表

本书是云计算技术与应用专业校企合作系列教材，开发了丰富的数字化教学资源，可使用的教学资源见表 2。

表 2　课程教学资源一览表

| 序　号 | 资 源 名 称 | 表现形式与内涵 |
|---|---|---|
| 1 | 课程简介 | Word 文档，包括对课程内容、项目设计、课时、适用对象等介绍，以便读者快速对本课程建立感性认识 |
| 2 | 学习指南 | Word 文档，包括对学前要求、学习目标要求以及学习路径和考核标准要求，为读者学习相关项目、实现相关任务提供参考与帮助 |
| 3 | 课程标准 | Word 文档，包括课程定位、课程目标要求以及课程内容与要求，为教师实施项目教学提供参考与帮助 |
| 4 | 整体设计 | Word 文档，包括课程设计思路、课程具体的目标要求以及课程内容设计和能力训练设计，同时给出考核方案设计，为教师实施项目教学提供参考与帮助 |

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



续表

| 序　号 | 资 源 名 称 | 表现形式与内涵 |
|---|---|---|
| 5 | PPT 和微课 | PPT 文件和 MP4 视频文件，为读者学习 Hadoop 大数据平台构建与应用的相关任务提供指导与帮助 |
| 6 | 教学单元设计 | Word 文档，分项目给出教学设计，帮助教师类读者完成项目的教学设计，为教师实施项目教学提供参考与帮助 |

| | | |
|---|---|---|
| 7 | 实训任务单 | Word 文档，为每个任务设计科学合理的实现步骤，为读者自行设计与实现相关任务提供指导与帮助 |
| 8 | 案例源码 | Tar 包，包括项目案例和综合案例，为任务的实现提供参考 |
| 9 | 习题库、试卷库 | Word 文档，习题涵盖必要的理论知识和技能操作，试卷包括单元测试和课程测试。通过练习和训练，巩固对相关知识的理解 |

本书通过二维码引入典型操作的微课视频，建议读者在 WiFi 环境下扫码观看。另外，本书配套的资源包、运行脚本、电子教案等，读者可登录 http://www.1daoyun.com 下载。

## 四、致谢

本书由南京交通职业技术学院的米洪、陈永任主编，张鸽、郑莹、史律任副主编，并与南京第五十五所技术开发有限公司、江苏一道云科技发展有限公司共同编写，贾俐俐、陈晓琴任主审。同时，在本书的编写过程中，参阅了国内外同行编写的相关著作和文献，谨向各位作者致以深深谢意！

由于作者水平有限，书中难免存在错误与疏漏之处，恳请广大读者批评、指正。

编　者

2017.10

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

# 目　录


项目 1　认识大数据，实现学情分析系统设计与环境搭建 …… 1
学习目标 …… 1
学习情境 …… 1
任务 1.1　设计学情分析系统 …… 3
任务描述 …… 3
任务目标 …… 3
知识准备 …… 3
任务实施 …… 17
项目实训 …… 19
任务 1.2　构建学情分析系统开发环境 …… 20
任务描述 …… 20
任务目标 …… 20
知识准备 …… 20
任务实施 …… 21
项目实训 …… 42
项目小结 …… 42
项目 2　Hadoop 大数据平台的构建 …… 43
学习目标 …… 43
学习情境 …… 43
任务 2.1　搭建 Ambari Hadoop 系统 …… 44
任务描述 …… 44
任务目标 …… 44
知识准备 …… 44
任务实施 …… 48
项目实训 …… 58
任务 2.2　使用 Apache Ambari 管理 Hadoop 集群 …… 58
任务描述 …… 58
任务目标 …… 58
知识准备 …… 58
任务实施 …… 59
项目实训 …… 64
项目小结 …… 65
项目 3　就业岗位数据的采集和存储 …… 67
学习目标 …… 67
学习情境 …… 67
任务 3.1　使用网络爬虫采集岗位招聘数据 …… 68
任务描述 …… 68
任务目标 …… 68
知识准备 …… 68
任务实施 …… 69
项目实训 …… 76
任务 3.2　将采集的数据上传到 HDFS …… 76
任务描述 …… 76
任务目标 …… 76
知识准备 …… 76
任务实施 …… 78
项目实训 …… 86
任务 3.3　使用 Sqoop 转换 MySQL 中的学生成绩数据到 Hive 中 …… 87
任务描述 …… 87
任务目标 …… 87
知识准备 …… 87
任务实施 …… 87
项目实训 …… 96
项目小结 …… 96
项目 4　岗位数据的处理 …… 97
学习目标 …… 97
学习情境 …… 97
任务 4.1　使用 Hadoop MR 对数据进行清洗 …… 98


非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！




任务描述 …… 98
任务目标 …… 98
知识准备 …… 98
任务实施 …… 99
项目实训 …… 105
任务 4.2 使用命令行对 Hive 进行数据查询和过滤 …… 105
任务描述 …… 105
任务目标 …… 105
知识准备 …… 106
任务实施 …… 106
项目实训 …… 108
任务 4.3 使用 Java API 对 Hive 进行连接与操作 …… 108
任务描述 …… 108
任务目标 …… 108
知识准备 …… 109
任务实施 …… 109
项目实训 …… 117
项目小结 …… 117
项目 5 岗位和技能数据的分析 …… 119
学习目标 …… 119
学习情境 …… 119
任务 5.1 使用 Mahout 命令进行岗位聚类分析 …… 120
任务描述 …… 120
任务目标 …… 120
知识准备 …… 120
任务实施 …… 126
项目实训 …… 144
任务 5.2 使用 Mahout 的 Java API 进行岗位分析聚类开发 …… 144
任务描述 …… 144
任务目标 …… 144
知识准备 …… 144
任务实施 …… 144
项目实训 …… 149
任务 5.3 使用 Mahout 推荐工作岗位 …… 149
任务描述 …… 149
任务目标 …… 149
知识准备 …… 149
任务实施 …… 151
项目实训 …… 156
项目小结 …… 157
项目 6 数据的可视化 …… 159
学习目标 …… 159
学习情境 …… 159
任务 6.1 使用 Excel 可视化数据 …… 160
任务描述 …… 160
任务目标 …… 160
知识准备 …… 160
任务实施 …… 160
项目实训 …… 162
任务 6.2 使用 ECharts 可视化数据 …… 162
任务描述 …… 162
任务目标 …… 162
知识准备 …… 163
任务实施 …… 163
项目实训 …… 174
任务 6.3 使用 D3.js 可视化数据 …… 174
任务描述 …… 174
任务目标 …… 174
知识准备 …… 174
任务实施 …… 175
项目实训 …… 179
项目小结 …… 179
参考文献 …… 181


非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

# 项目 1

# 认识大数据，实现学情分析系统设计与环境搭建

PPT1：认识大数据，实现学情分析系统设计与环境搭建

## 学习目标

【知识目标】

- 掌握大数据的概念和特征。
- 了解大数据处理与分析流程。
- 学会学情分析系统架构的设计方法。

【技能目标】

- 熟悉大数据分析与处理工具的使用。
- 学会大数据开发环境的配置操作。

## 学习情境

2012 年，联合国在发布的《大数据促发展：挑战与机遇》白皮书中指出：“大数据时代已经到来，大数据的出现将会对社会各个领域产生深刻影响。”大数据正在实现人类工作、生活与思维的大变革，其“威力”也强烈地冲击着整个教育系统，正在成为推动教育系统创新与变革的颠覆性力量。

目前大数据在教育领域的应用还存在诸多挑战，学校业务部门没有清晰的大数据需求导致数据资产逐渐流失；院校内部数据孤岛严重，导致数据价值不能被充分挖掘；数据可用性低、质量差，导致数据无法被利用等，诸多因素制约了大数据在教育领域的应用。

米老师是一名高校教师，基于对大数据的认识和了解，他想探索“大数据”在高校教育领域的应用，想通过收集学生就业趋向、学习兴趣、专业技能、岗位需求的数据，结合“因材施教”教育理论，帮助学生认识自己，结合学生实际情况，为其推荐学习资源、就业信息等，如图 1-1 所示。为了实现这一目标，本项目主要学习大数据的概念、相关的技术和应用，使学生对大数据的相关知识和产业链有一个清晰的认识。

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



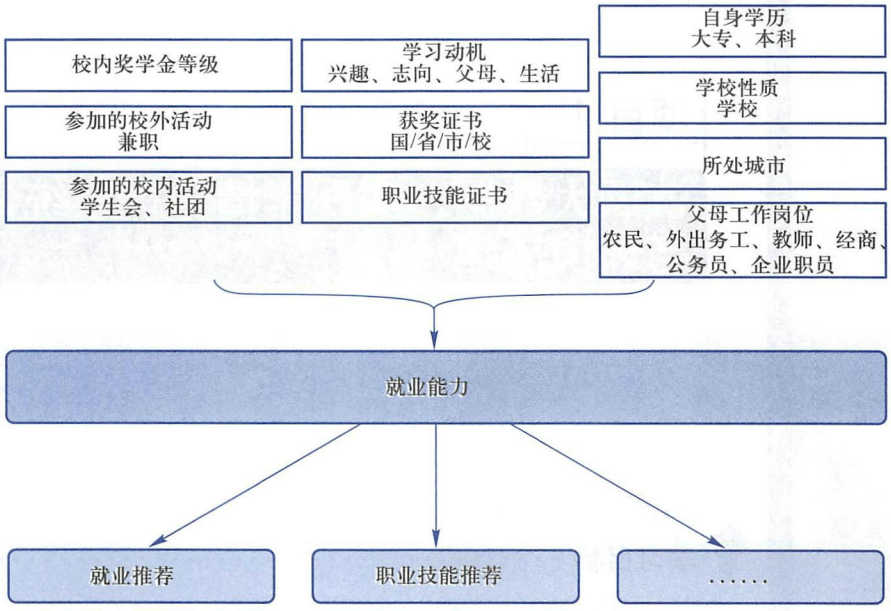



图 1-1 就业关系影响图

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



# 任务 1.1　设计学情分析系统

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习大数据导论等相关知识内容，熟悉大数据的定义、大数据的基本特征及大数据处理与分析的相关技术、工具或产品等。

2. 明确学情分析系统的设计目标和实现途径，理清学情分析系统所要采集的数据信息，并进一步关注数据信息的格式，进而思考数据导入和预处理的方法及目标、数据统计与分析手段和数据挖掘的目标，为学情分析系统目标的实现奠定基础。

3. 完成学情分析系统的设计。

## 任务目标

1. 熟悉大数据的概念和特征。
2. 熟悉大数据处理与分析流程和工具使用。
3. 熟悉学情分析系统的架构设计方法。

## 知识准备

### 1. 大数据背景知识

微课：1-1 认识大数据

大数据已成为时下 IT 界最炙手可热的名词，在全球引领了新一轮数据技术革命的浪潮。经过 2012 年的蓄势待发，2013 年被称为世界大数据元年，标志着世界正式步入了大数据时代。

根据支付宝官方发布的数据，2016 年“双十一”狂欢节总交易额超 1,207 亿元，按照这个金额计算平均每分钟交易额约为 8,382 万元。根据淘宝数据显示，在“双十一”当天淘宝活跃用户超过 1 亿，平均每分钟的活跃用户超过 69,444 个。目前，每分钟之内会有 4,310 人访问亚马逊（Amazon）网站；优步（Uber）每分钟能获得 694 个订单；苹果（Apple）用户每分钟会下载 51,000 个应用；YouTube 用户每分钟会上传 300 小时的新视频；Netflix 用户每分钟之内会观看 77,160 小时的视频。在 2015 年世界互联网大会上，腾讯副总裁赖智明表示，微信红包一天的收发量是 22 亿个，平均每分钟红包收发量是 1,527,777 个。每分钟谷歌（Google）搜索引擎的搜索量可达 278 万次；脸书（Facebook）用户每分钟点赞 4,166,667 次。

用适当的统计分析方法对收集来的大量数据进行分析，将它们加以汇总和理解，以求最大化地开发数据的功能，发挥数据的作用。Hadoop 从某些方面来讲，与大数据结合紧密，它最擅长的事情就是可以高效地处理海量规模的数据，因此可以说它就是为大数据而生的。

想要系统地认知大数据，必须要全面而细致地分解它，接下来将从 3 个层面展开

严禁网络传播本PDF，违者责任自负！

介绍，如图 1-2 所示。

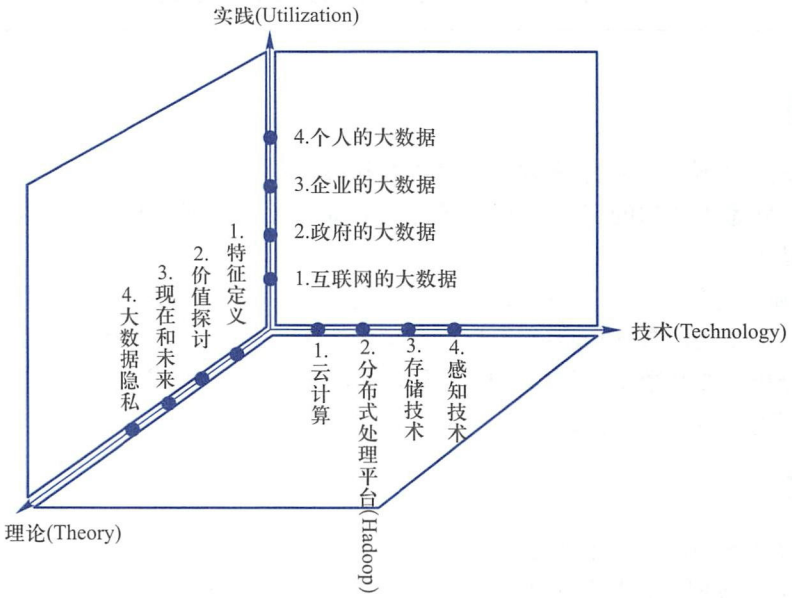



图 1-2 大数据的 3 个层面

第 1 层面是理论。理论是认知的必经途径，也是被广泛认同和传播的基线。从大数据的特征定义理解行业对大数据的整体描绘和定性；从对大数据价值的探讨来深入解析大数据的珍贵所在；从对大数据的现在和未来去洞悉大数据的发展趋势；从大数据隐私这个特别而重要的视角审视人和数据之间的长久博弈。

第 2 层面是技术。技术是大数据价值体现的手段和前进的基石，分别从云计算、分布式处理技术、存储技术和感知技术的发展来说明大数据从采集、处理、存储到形成结果的整个过程。

笔记

第 3 层面是实践。实践是大数据的最终价值体现，分别从互联网的大数据、政府的大数据、企业的大数据和个人的大数据 4 个方面来描绘大数据已经展现的美好景象以及即将实现的蓝图。

（1）理论层面认知大数据

1）大数据的特征定义。最早提出大数据时代到来的是麦肯锡公司，它是美国首屈一指的咨询公司，是研究大数据的先驱。在其报告 *Big data: The next frontier for innovation, competition, and productivity* 中给出的大数据定义是：大数据指的是大小超出常规的数据库工具获取、存储、管理和分析能力的数据集。

业界（IBM 最早定义）将大数据的特征归纳为以下 4 个“V”：

① 数据体量巨大（Volume）。大数据的起始计量单位至少是 PB（1,024 TB）、EB（1,024 PB）或 ZB（1,024 EB）。

② 数据类型繁多（Variety）。如网络日志、视频、图片、地理位置信息等。

③ 价值密度低，商业价值高（Value）。由于数据采集的不及时，数据样本不全

面、数据不连续等，可能会导致数据失真。但当数据量达到一定规模时，可以通过更多的数据达到更真实全面的反馈。

④ 处理速度快（Velocity）。在大数据时代要用大数据思维去发掘大数据的潜在价值。

本书认同大数据研究机构 Gartner 给出的定义：大数据是指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能处理具有更强决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。

2）大数据的现在和未来。现在，大数据应用价值已在各行各业凸显，大数据帮助政府实现市场经济调控、公共卫生安全防范、灾难预警、社会舆论监督；大数据帮助城市预防犯罪、实现智慧交通、提升紧急应急能力；大数据帮助医疗机构建立患者的疾病风险跟踪机制、帮助医药企业提升药品的临床使用效果、帮助艾滋病研究机构为患者提供定制的药物；大数据帮助航空公司节省运营成本、帮助电信企业实现售后服务质量提升、帮助保险企业识别欺诈骗保行为、帮助快递公司监测分析运输车辆的故障险情以便提前预警维修、帮助电力公司有效识别预警即将发生故障的设备。

不管大数据的核心价值是不是预测，但是基于大数据形成决策的模式已经为不少的企业带来了盈利和声誉。从大数据的价值链条来分析，存在以下 3 种模式：

① 手握大数据，但是没有利用好。比较典型的是金融机构、电信行业、政府机构等。

② 没有数据，但是知道如何帮助有数据的用户利用它。比较典型的是 IT 咨询和服务企业，如埃森哲（Accenture）、IBM、甲骨文（Oracle）等公司。

③ 既有数据，又有大数据思维。比较典型的是谷歌公司、亚马逊公司、万事达公司（Mastercard）等。

未来在大数据领域最具有价值的是以下两种：

① 拥有大数据思维的人，这种人可以将大数据的潜在价值转化为实际利益。

② 还未被大数据触及过的业务领域。这些是还未被挖掘的金矿，即所谓的蓝海。

3）大数据带来的隐私问题。大数据时代，隐私保护是必须面对的问题，当在不同的网站上注册了个人信息后，可能这些信息已经被扩散出去了。当人们莫名其妙地接到各种邮件、电话、短信的滋扰时，不会想到自己的电话号码、邮箱、生日、购买记录、收入水平、家庭住址、亲朋好友等私人信息早就被各种商业机构非法存储或贱卖给其他任何有需要的企业或个人。当微博、微信、QQ 空间这些社交平台肆意地吞噬着数亿用户的各种信息时，就很难指望隐私权得到保护。即使用户在某个地方删除了相关信息，也许这些信息已经被其他人转载或保存，更有可能已经被存为网页快照，早已提供给任意用户搜索。因此在大数据的背景下，很多人都在积极抵制无底线的数字化，这种大数据和个体之间的博弈还会一直继续下去。

当很多互联网企业意识到隐私对于用户的重要性时，为了继续得到用户的信任，采取了很多办法，如谷歌公司承诺保留用户的搜索记录时间为 9 个月，有的浏览器厂商提供了无痕冲浪模式，还有社交网站拒绝公共搜索引擎的爬虫，并将提供出去的数

据全部采取匿名方式处理等。

被誉为“大数据商业应用第一人”的维克托·迈尔·舍恩伯格在《大数据时代》一书中给予了人们一些如何有效保护大数据背景下隐私权的建议：

① 减少信息的数字化。

② 建立隐私权立法。

③ 增强数字隐私权基础设施（类似 DRM 数字版权管理）。

④ 改变人类认知（接受忽略过去）。

⑤ 创造良性的信息生态。

⑥ 完全语境化。

（2）技术层面认知大数据

1）云技术。大数据常和云计算联系到一起，因为实时的大型数据集分析需要分布式处理框架来向数十、数百甚至数万的计算机分配工作。它的特色在于对海量数据的挖掘。如今，在谷歌、亚马逊、脸书等一批互联网企业引领下，创建了一种行之有效的模式，即云计算提供基础架构平台，大数据应用可以运行在这个平台上。

业内认为两者的关系为：没有大数据的信息积淀，则云计算的计算能力再强大，也难以找到用武之地；没有云计算的处理能力，则大数据的信息积淀再丰富，也终究只是镜花水月。

云计算和大数据之间的关系如图 1-3 所示。

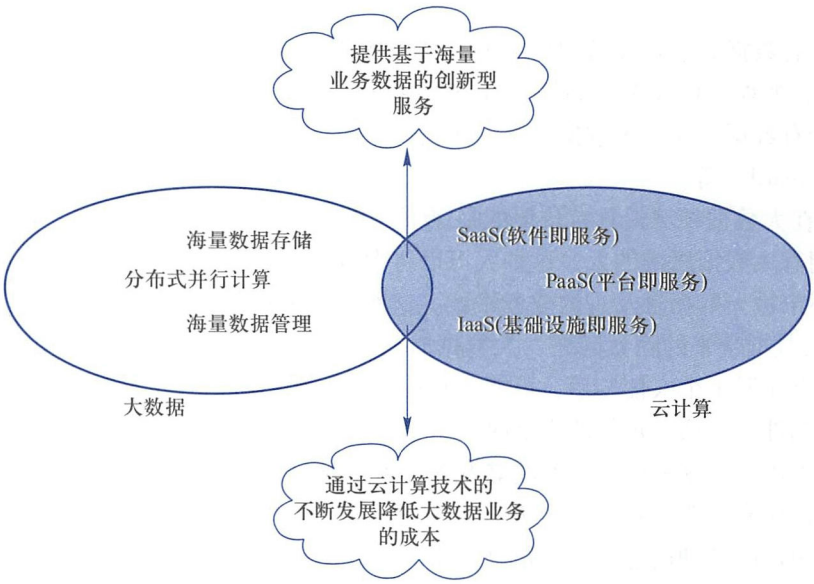



图 1-3 云计算与大数据的关系

两者之间结合后会产生如下效应：可以提供更多基于海量业务数据的创新型服务，并通过云计算技术的不断发展降低大数据业务的创新成本。云计算与大数据最明显的区分在以下两个方面：

① 在概念上两者有所不同。云计算改变了 IT，而大数据则改变了业务。然而大

数据必须有云作为基础架构，才能得以顺畅运营。

② 大数据和云计算的目标受众不同，云计算是CIO等关心的技术层，是一个进阶的IT解决方案。而大数据是CEO关注的、是业务层的产品，而大数据的决策者是业务层。

笔记

2）分布式处理技术。分布式处理系统可以将不同地点、具有不同功能、拥有不同数据的多台计算机用通信网络连接起来，在控制系统的统一管理下，协调地完成信息处理任务。

大数据分布式处理系统的典型代表是Hadoop。它包含一个MapReduce软件框架，能以一种可靠、高效、可伸缩的方式对大数据进行分布式处理。MapReduce是谷歌提出的一种云计算的核心计算模式，是一种分布式运算技术，也是简化的分布式编程模式。MapReduce模式的主要思想是将要执行的问题（如程序）拆解成Map（映射）和Reduce（规约）的方式，在数据被分割后通过Map函数的程序将数据映射成不同的区块，分配给计算机集群处理达到分布式运算的效果，再通过Reduce函数的程序将结果汇整，从而输出开发者需要的结果。

3）存储技术。大数据可以抽象地分为大数据存储和大数据分析，这两者的关系是大数据存储的目的是支撑大数据分析。到目前为止，大数据存储和大数据分析是两种截然不同的计算机技术领域：大数据存储致力于研发可以扩展至PB甚至EB级别的数据存储平台；大数据分析关注在最短时间内处理大量不同类型的数据集。

大数据存储服务商的典型代表是亚马逊公司，其S3是一种面向Internet的存储服务，旨在让开发人员能更轻松地进行网络规模计算。亚马逊的S3提供一个简明的Web服务界面，用户可以在任何地点通过访问该界面，存储和检索任意大小的数据。此服务让所有开发人员都能访问同一个具备高扩展性、高可靠性、高安全性和高性价比的基础设施，亚马逊就利用它来运行其全球的网站网络。亚马逊的S3云的存储对象已达到万亿级别，而且性能表现相当良好，目前已经拥有万亿跨地域存储对象，同时亚马逊提供的专业云计算服务Amazon Web Services（简称AWS）的对象执行请求也达到百万的峰值数量。目前全球范围内已经有数以十万计的企业在通过AWS运行自己的全部或者部分日常业务。这些企业用户遍布190多个国家，几乎世界上的每个角落都有亚马逊用户的身影。

4）感知技术。大数据的采集和感知技术的发展是紧密联系的。以传感器技术、指纹识别技术、RFID技术、坐标定位技术等为基础的感知能力提升同样是物联网发展的基石。全世界的工业设备、汽车、电表上有着无数的数码传感器，随时测量和传递着有关位置、运动、震动、温度、湿度乃至空气中化学物质的变化，都会产生海量的数据信息。

而随着智能手机的普及，感知技术可谓迎来了发展的高峰期，除了地理位置信息被广泛地应用外，一些新的感知手段也开始登上舞台，如iPhone 7在Home键内镶嵌指纹传感器，新型手机可通过呼气直接检测燃烧脂肪量，即将面世的手机的嗅觉传感器可以监测从空气污染到危险的化学药品，微软公司正在研发可感知用户当前心情的智能手机技术，谷歌眼镜InSight的新技术可通过衣着进行人物识别等。

其实，这些感知被逐渐捕获的过程就是世界被数据化的过程，一旦世界被完全数据化了，那么世界的本质也就是信息了。

所以说“人类以前延续的是文明，现在传承的是信息。”

（3）实践层面认知大数据

1）互联网的大数据。互联网上的数据每年增长 50%，每两年便将翻一番，而目前世界上 90%以上的数据是最近几年才产生的。据互联网数据中心（IDC）预测，到 2020 年全球将总共拥有 35 ZB 的数据量。互联网是大数据发展的前哨阵地，随着 Web 2.0 时代的发展，人们似乎都习惯了将自己的生活通过网络进行数据化，方便分享以及记录并回忆。

互联网上的大数据很难清晰地界定分类界限。以下是中国互联网公司三巨头百度公司、阿里巴巴集团、腾讯公司（简称 BAT）的大数据。

百度公司拥有用户搜索表征的需求数据、爬虫和阿拉丁获取的公共 Web 数据两种类型的大数据。其通过对网页数据的爬取、网页内容的组织和解析，并进行语义分析，进而产生对搜索需求的精准理解，以便从海量数据中找准结果，实质上就是一个数据的获取、组织、分析和挖掘的过程。搜索引擎在大数据时代面临的挑战有更多的暗网数据、更多的 Web 化但是没有结构化的数据和更多的 Web 化、结构化但是封闭的数据。

阿里巴巴拥有交易数据和信用数据，这两种数据更容易变现，挖掘出商业价值。除此之外，阿里巴巴还通过投资等方式掌握了部分社交数据、移动数据，如微博社交数据和高德地图相关数据等。

腾讯公司拥有用户关系数据和基于此产生的社交数据。这些数据可以分析人们的生活和行为，从中挖掘出政治、社会、文化、商业、健康等领域的信息，甚至预测未来。

在信息技术更为发达的美国，除了行业知名的谷歌公司和脸书公司外，已经涌现了很多专门经营大数据类型产品的公司。

Metamarkets：该公司对推特网站（Twitter）用户的支付、签到和一些与互联网相关的问题进行了分析，为客户提供很好的数据分析服务。

Tableau：该公司主要集中于将海量数据以可视化的方式展现出来，为数字媒体提供了一个新的展示数据的方式。Tableau 为用户提供了一个免费工具，任何人在没有编程知识背景的情况下都能制造出数据专用图表。该软件还能对数据进行分析，并提供有价值的建议。

ParAccel：该公司向美国执法机构提供数据分析，如对 15,000 个有犯罪前科的人进行跟踪，从而向执法机构提供了参考性较高的犯罪预测，因此也被称为“犯罪的预言者”。

QlikTech：该公司旗下的 Qlikview 是一个商业智能领域的自主服务工具，能够应用于科学研究和艺术等领域。为了帮助开发者对这些数据进行分析，QlikTech 提供了对原始数据进行可视化处理等功能的工具。

GoodData：该公司希望帮助客户从数据中挖掘财富，主要面向商业用户和 IT 企业

高管，提供数据存储、性能报告、数据分析等工具。

简要归纳，互联网中大数据的典型代表包括以下各个方面。

① 用户行为数据：精准广告投放、内容推荐、行为习惯和喜好分析、产品优化等。

② 用户消费数据：精准营销、信用记录分析、活动促销、理财等。

③ 用户地理位置数据：在线离线/线上到线下推广（简称 O2O）、商家推荐、交友推荐等。

④ 互联网金融数据：点对点网络借款（简称 P2P）、小额贷款、支付、信用、供应链金融等。

⑤ 用户社交数据：潮流趋势分析、流行元素分析、受欢迎程度分析、舆论监控分析、社会问题分析等。

2）政府的大数据。美国政府曾宣布投资 2 亿美元拉动大数据相关产业发展，将“大数据战略”上升为国家意志。美国政府将数据定义为“未来的新石油”，并表示一个国家拥有数据的规模、活性及解释运用的能力将成为综合国力的重要组成部分。未来，对数据的占有和控制甚至将成为陆权、海权、空权之外的另一种国家核心资产。

在国内，政府各个部门都握有构成社会基础的原始数据，如气象数据、金融数据、信用数据、电力数据、煤气数据、自来水数据、道路交通数据、客运数据、安全刑事案件数据等。这些数据在每个政府部门里面看起来是单一的、静态的。但是，如果政府可以将这些数据关联起来，并对这些数据进行有效的关联分析和统一管理，这些数据必定将获得新生，其价值是无法估量的。

具体来说，现在城市都在走向智能化，如智能电网、智慧交通、智慧医疗、智慧环保、智慧城市等，这些都依托于大数据，可以说大数据是智慧的核心能源。从国内整体投资规模来看，自 2012 年住建部发布《关于开展国家智慧城市试点工作的通知》至今，全国开建智慧城市的城市数超过 290 个，通信网络和数据平台等基础设施建设投资规模接近 5,000 亿元。大数据为智慧城市的各个领域提供决策支持：在城市规划方面，通过对城市地理、气象等自然信息和经济、社会、文化、人口等人文社会信息的挖掘，可以为城市规划提供决策，强化城市管理服务的科学性和前瞻性；在交通管理方面，通过对道路交通信息的实时挖掘，能有效缓解交通拥堵，并快速响应突发状况，为城市交通的良性运转提供科学的决策依据；在舆情监控方面，通过网络关键词搜索及语义智能分析，能提高舆情分析的及时性、全面性，全面掌握社情民意，提高公共服务能力，应对网络突发的公共事件，打击违法犯罪；在安防与防灾领域，通过大数据的挖掘，可以及时发现人为或自然灾害、恐怖事件，提高应急处理能力和安全防范能力。

另外，作为国家的管理者，政府应该有勇气将手中的数据逐步开放，供给更多有能力的机构组织或个人来分析并加以利用，以加速造福人类。例如，美国政府就筹建了一个 Data. gov 网站，这个重要举措是实现政府机构的数据公开。

3）企业的大数据。电商企业“首席惊喜官”（CXO）们最关注的还是报表曲线

的背后具有的信息，他们该做怎样的决策，这一切都需要通过数据来传递和支撑。在理想的世界中，大数据是巨大的杠杆，可以改变公司的影响力，带来竞争差异、节省费用、增加利润、愉悦买家、奖赏忠诚用户、将潜在客户转化为客户、增加吸引力、打败竞争对手、开拓用户群并创造市场。

现实生活中，有一部分行业的传统企业最需要大数据服务，如对大量消费者提供产品或服务的企业（精准营销）、做小而美模式的中长尾企业（服务转型）、面临互联网压力之下必须转型的传统企业（生死存亡）。

对于企业的大数据，还有一种预测：随着数据逐渐成为企业的一种资产，数据产业会向传统企业的供应链模式发展，最终形成“数据供应链”。这其中有两个明显的现象：

① 外部数据的重要性日益超过内部数据。在互联互通的网络时代，单一企业的内部数据与整个互联网数据相比只是沧海一粟。

② 能提供包括数据供应、数据整合与加工、数据应用等多环节服务的公司会有明显的综合竞争优势。

对于提供大数据服务的企业来说，等待的是合作机会，就像微软公司的史密斯说“给我提供一些数据，我就能做一些改变。如果给我提供所有数据，我就能拯救世界。”

如今，一直做企业服务的巨头公司的优势将不复存在，随着模仿型新兴互联网企业加入战局，开启了残酷竞争模式。为何会出现这种局面？从 IT 产业的发展来看，第一代 IT 巨头大多是 ToB（即面向企业）的，如 IBM、微软、甲骨文、SAP、惠普（HP）这类传统 IT 企业。第二代 IT 巨头大多是 ToC（即面向用户）的，如雅虎、谷歌、亚马逊、脸书这类互联网企业。大数据到来前，这两类公司彼此之间基本是井水不犯河水，但在当前这个大数据时代，这两类公司已经开始直接竞争，如亚马逊已经开始提供云模式的数据仓库服务，直接抢占 IBM、甲骨文的市场。该现象出现的本质原因是，在互联网巨头的带动下，传统 IT 巨头的客户普遍开始从事电子商务业务，正是由于客户进入了互联网，所以传统 IT 巨头们不情愿地被拖入了互联网领域。如果他们不进入互联网，他们的业务必将萎缩。在进入互联网后，他们又必须将云技术、大数据等互联网最具有优势的技术通过封装打造成自己的产品再提供给企业。

笔记

以 IBM 为例，上一个十年，他们抛弃了计算机硬件，成功转向了软件和服务。目前他们将远离服务与咨询，更多地专注于因大数据分析软件而带来的全新业务增长点。IBM 执行总裁罗睿兰认为“数据将成为一切行业当中决定胜负的根本因素，最终数据将成为人类至关重要的自然资源。”IBM 积极地提出了“大数据平台”架构。该平台的四大核心能力包括 Hadoop 系统、流计算（Stream Computing）、数据仓库（Data Warehouse）和信息整合与治理（Information Integration and Governance），如图 1-4 所示。

另外一家亟待通过云和大数据战略而复苏的巨头惠普公司也推出了自己的产品 HAVEn，它是一个可以自由扩展伸缩的大数据解决方案。该解决方案由惠普智能数据操作层（Autonomy IDOL）、惠普分析平台（HP Vertica）、惠普企业安全管理（HP

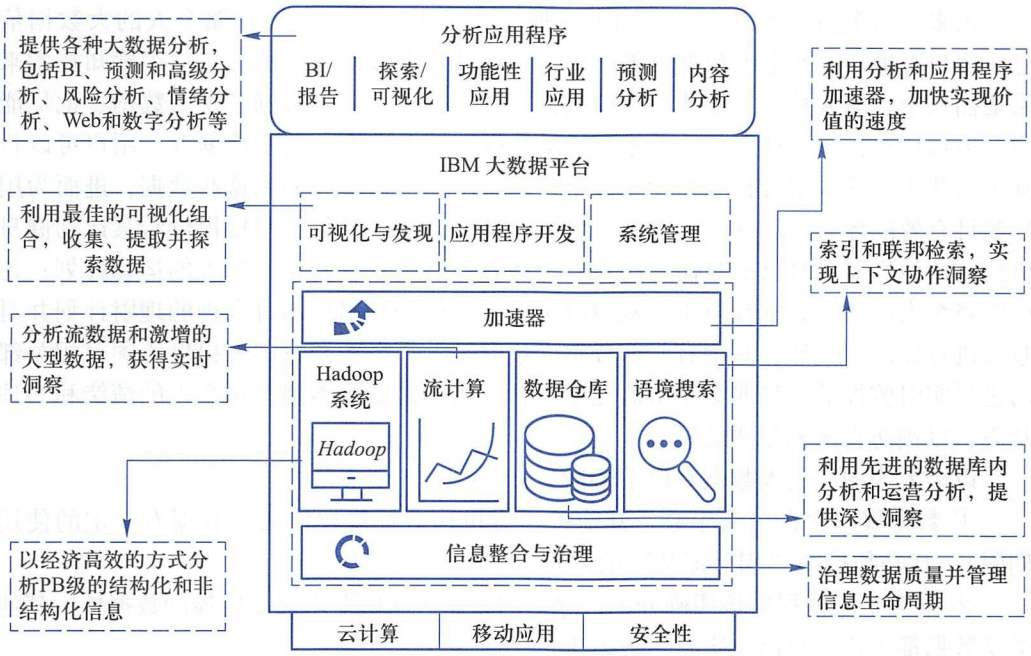



图 1-4　IBM 大数据平台和应用程序框架

ArcSight）和惠普运营管理（HP Operations Management）四大技术组成，同时还支持 Hadoop 这样的通用技术。HAVEn 不是一个软件平台，而是一个生态环境，四大组成部分满足不同的应用场景需要：Autonomy 解决音视频识别的方案；Vertica 解决数据处理的速度和效率的方案；ArcSight 解决机器的记录信息处理，帮助企业获得更高安全级别的管理；Operations Management 解决的不仅是外部数据的处理，而且包括了 IT 基础设施产生的数据，如图 1-5 所示。

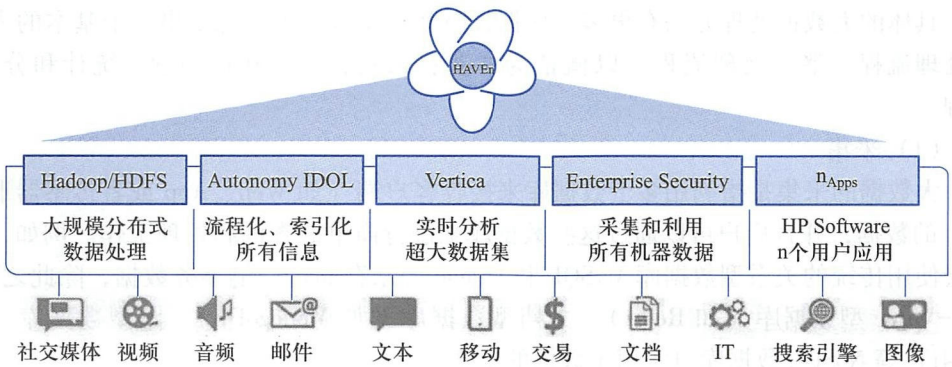



图 1-5　惠普公司的 HAVEn 大数据平台

4）个人的大数据。个人的大数据概念很少有人提及，简单来说，就是与个人相关联的各种有价值数据信息被有效采集后，可由本人授权提供第三方进行处理和使用，并获得第三方提供的数据服务。

未来，每个用户可以在互联网上注册个人的数据中心，以存储个人的大数据信息。用户可确定哪些个人数据可被采集，并通过可穿戴设备或植入芯片等感知技术来采集捕获个人的大数据，如牙齿监控数据、心率数据、体温数据、视力数据、记忆能力、地理位置信息、社会关系数据、运动数据、饮食数据、购物数据等。用户可以将其中的牙齿监测数据授权给某牙科诊所使用，由他们监控和使用这些数据，进而为用户制订有效的牙齿防治和维护计划；也可以将个人的运动数据授权提供给某运动健身机构，由他们监测自己的身体运动机能，并有针对地制订和调整个人的运动计划；还可以将个人的消费数据授权给金融理财机构，由他们帮用户制订合理的理财计划并对收益进行预测。当然，其中有一部分个人数据是无须个人授权即可提供给国家相关部门进行实时监控的，如罪案预防监控中心可以实时地监控本地区每个人的情绪和心理状态，以预防自杀和犯罪的发生。

以个人为中心的大数据有以下特性：

① 数据仅留存在个人中心，其他第三方机构只被授权使用（数据有一定的使用期限），且必须接受“用后即焚”的监管。

② 采集个人数据应该明确分类，除了国家立法明确要求接受监控的数据外，其他类型数据都由用户自己决定是否被采集。

③ 数据的使用只能由用户进行授权，数据中心可帮助用户监控个人数据的整个生命周期。

展望过于美好，也许实现个人数据中心的愿望还很遥远，也许这还不是解决个人数据隐私的最好方法，也许业界对大数据的无限渴求会阻止个人数据中心的实现，但是随着数据越来越多，在缺乏监管之后，必然会有一场激烈的博弈——到底是数据重要还是隐私重要，以及是以商业为中心还是以个人为中心，让我们拭目以待。

笔记

### 2. 大数据处理流程

具体的大数据处理方法有很多，根据长时间的实践，可以总结出一个基本的大数据处理流程。整个处理流程可以概括为 4 步：采集、导入和预处理、统计和分析、挖掘。

（1）采集

大数据的采集是指利用多个数据库来接收客户端（如 Web、App 或者传感器形式等）的数据，并且用户可以通过这些数据库来进行简单的查询和处理工作。例如，电商会使用传统的关系型数据库 MySQL 和 Oracle 等来存储每一笔事务数据，除此之外，Key-Value 型数据库（如 Redis）、文档型数据库（如 MonogoDB）、图型数据库（如 Neo4j）等 NoSQL 数据库也常用于数据的采集。

在大数据的采集过程中，其主要特点和挑战是并发数高，因为同时有可能会有成千上万的用户来进行访问和操作，如火车票售票网站和淘宝，它们并发的访问量在峰值时达到上百万，需要在采集端部署大量数据库才能支撑，且如何在这些数据库之间进行负载均衡和分片是需要深入的思考和设计的。

（2）导入和预处理

采集端本身会有很多数据库，如果要对这些海量数据进行有效的分析，还是应该将这些来自前端的数据导入到一个集中的大型分布式数据库（SequoiaDB）或者分布式存储集群（OneProxy）中，并且可以在导入基础上做一些简单的清洗和预处理工作。也有一些用户会在导入时使用来自 Twitter、LinkedIn 等公司相继开源的流式计算系统 Storm、分布式发布订阅消息系统 Kafka、雅虎之前开源的 S4 等对数据进行流式计算，来满足部分业务的实时计算需求。

导入与预处理过程的特点和挑战主要是导入的数据量大，每秒的导入量经常会达到百兆，甚至千兆级别。

（3）统计与分析

统计与分析主要利用分布式数据库，或者分布式计算集群来对存储于其内的海量数据进行普通的分析和分类汇总等，以满足大多数常见的分析需求。在这些方面，一些实时性需求会用到易安信（EMC，一家美国信息存储资讯科技公司）的分布式数据库 GreenPlum、Oracle 的新一代数据库云服务器 Exadata 以及基于 MySQL 的列式存储 Infobright 等，而一些批处理或者基于半结构化数据的需求可以使用 Hadoop。

统计与分析这部分的主要特点和挑战是分析涉及的数据量大，其对系统资源，特别是 I/O 会有极大的占用。

（4）挖掘

与统计和分析过程不同的是，数据挖掘一般没有什么预先设定好的主题，主要是在现有数据上面进行基于各种算法的计算，从而起到预测（Predict）的效果，以便实现一些高级别数据分析的需求。比较典型算法有用于聚类的 K-Means、用于统计学习的 SVM 和用于分类的 Naive Bayes，主要使用的工具有 Hadoop 的 Mahout 等。

该过程的特点和挑战主要是用于挖掘的算法很复杂，并且计算涉及的数据量和计算量都很大，另外，常用数据挖掘算法都以单线程为主。

数据来自各个方面，在面对庞大而复杂的大数据，选择一个合适的处理工具显得很有必要。工欲善其事，必先利其器，一个好的工具不仅可以使工作事半功倍，也可以让人们在竞争日益激烈的云计算时代，挖掘大数据价值，及时调整战略方向。

### 3. 大数据分析工具

（1）Apache Hadoop

Hadoop 是一个能够对大量数据进行分布式处理的软件框架，并且 Hadoop 是以一种可靠、高效、可伸缩的方式进行数据处理的。Hadoop 的可靠体现在，假设计算元素和存储会失败，由于它维护多个工作数据副本，能够确保针对失败的节点重新分布处理。Hadoop 是高效的，因为它以并行的方式工作，通过并行处理加快处理速度。Hadoop 是可伸缩的，其能够处理 PB 级数据。此外，Hadoop 依赖于社区服务器，因此其成本比较低，任何人都可以使用，如图 1-6 所示。

Hadoop 是一个能够让用户轻松构建和使用的分布式计算平台。用户可以轻松地在 Hadoop 上开发和运行处理海量数据的应用程序。其具有以下主要优点。

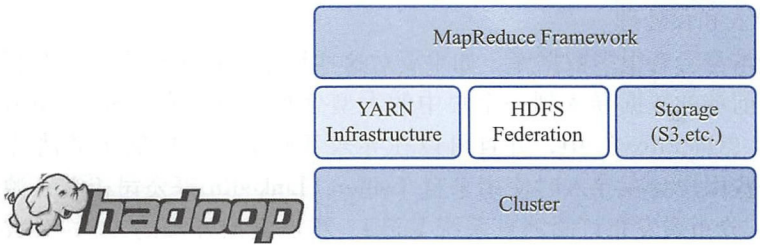



图 1-6 Hadoop 开源项目

① 高可靠性：Hadoop 按位存储和处理数据的能力值得信赖。

② 高扩展性：Hadoop 是在可用的计算机集群间分配数据并完成计算任务的，这些集群可以方便地扩展到数以千计的节点中。

③ 高效性：Hadoop 能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。

④ 高容错性：Hadoop 能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。

⑤ 平台与语言的选择灵活：Hadoop 带有用 Java 语言编写的框架，因此运行在 Linux 生产平台上是非常理想的。Hadoop 上的应用程序也可以使用其他语言编写，如 C++。

（2）Apache Spark

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark 是由加州大学伯克利分校（UCB）的 AMP 实验室所开源的类 Hadoop MapReduce 的通用并行框架，如图 1-7 所示。

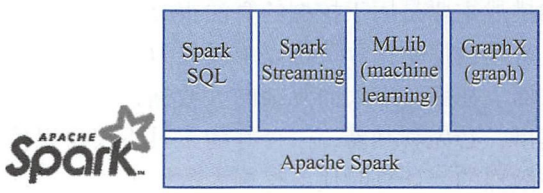



图 1-7 Apache Spark 开源项目

Spark 是在 Scala 语言中实现的，它将 Scala 用作其应用程序框架。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。

尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对 Hadoop 的补充，可以在 Hadoop 文件系统中并行运行。通过名为 Mesos 的第三方集群框架可以支持此行为。

Spark 性能特点包括以下几个方面。

① 高性能：内存计算下，Spark 比 MapReduce 快 100 倍。

② 易用性：Spark 提供了 80 多个高级运算符。

③ 通用性：Spark 提供了大量的库，包括 SQL、DataFrames、MLlib、GraphX、Spark Streaming，开发者可以在同一个应用程序中无缝组合使用这些库。

Spark 组成包括以下几个方面。

① Spark Core：包含 Spark 的基本功能，尤其是定义弹性数据集（RDD）的应用程序编程接口（API）、操作以及这两者上的动作。其他 Spark 的库都是构建在 RDD 和 Spark Core 之上的。

② Spark SQL：提供通过 Apache Hive 的 SQL 变体 Hive 查询语言（HiveQL）与 Spark 进行交互的 API。每个数据库表被当作一个 RDD，Spark SQL 查询被转换为 Spark 操作。

③ Spark Streaming：对实时数据流进行处理和控制。Spark Streaming 允许程序能够像普通 RDD 一样处理实时数据。

④ MLlib：一个常用机器学习算法库，算法被实现为对 RDD 的 Spark 操作。该库包含可扩展的学习算法，如分类、回归等需要对大量数据集进行迭代的操作。

⑤ GraphX：控制图、并行图操作、计算的一组算法和工具的集合。GraphX 扩展了 RDD API，包含控制图、创建子图、访问路径上所有顶点的操作。

（3）Apache Storm

Storm 由 Twitter 开源而来，是自由的开源软件，是一个分布式的、容错的实时计算系统。Storm 可以非常可靠地处理庞大的数据流，用于处理 Hadoop 的批量数据。Storm 很简单，支持许多种编程语言，使用起来非常有趣，如图 1-8 所示。

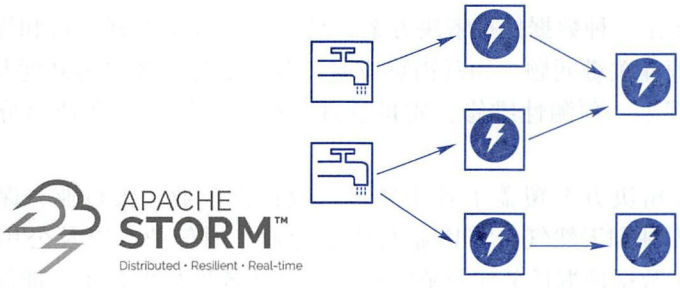



图 1-8　Apache Storm 开源项目

Storm 有许多应用领域，如实时分析、在线机器学习、不停顿的计算、分布式远过程调用（RPC，一种通过网络从远程计算机程序上请求服务的协议）、数据抽取、转换和加载（即 ETL）等。Storm 的处理速度惊人，经测试，每个节点每秒钟可以处理 100 万个数据元组。

（4）Apache Drill

为了帮助企业用户寻找更有效、更快的 Hadoop 数据查询的方法，Apache 软件基金会发起了名为 Drill 的开源项目。Apache Drill 实现了谷歌的交互式数据分析系统 Dremel，如图 1-9 所示。

据 Hadoop 厂商 MapR Technologies 公司产品经理 Tomer Shiran 介绍，Drill 已经作为 Apache 孵化器项目来运作，将面向全球软件工程师持续推广。

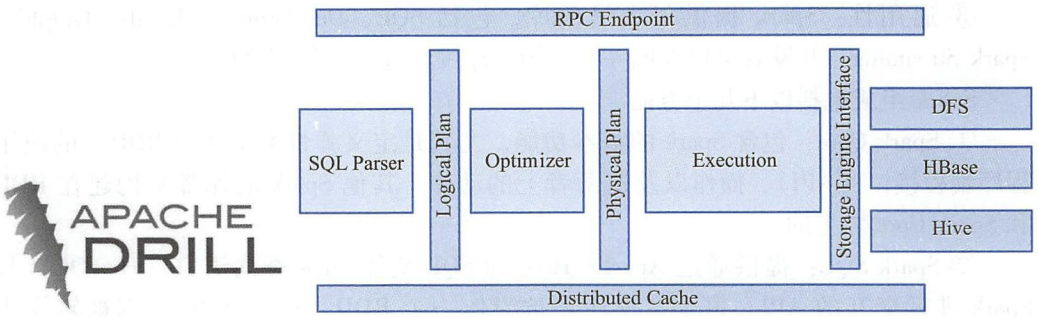



图 1-9 Apache Drill 开源项目

该项目将会创建出开源版本的谷歌 Dremel Hadoop 工具（谷歌使用该工具来为 Hadoop 数据分析工具的互联网应用提速）。而 Drill 将有助于 Hadoop 用户实现更快查询海量数据集的目的。

Drill 项目其实也是从谷歌的 Dremel 项目中获得灵感。该项目帮助谷歌实现海量数据集的分析处理，包括分析抓取 Web 文档、跟踪安装在 Google Play（一个由谷歌为 Android 设备开发的在线应用程序商店）上的应用程序数据、分析垃圾邮件、分析谷歌分布式构建系统上的测试结果等。

通过开发 Drill 开源项目，组织机构将有望建立 Drill 所属的 API 接口和灵活强大的体系架构，从而帮助支持广泛的数据源、数据格式和查询语言。

（5）RapidMiner

RapidMiner 是一种数据挖掘解决方案，具有丰富数据挖掘分析和算法功能，常用于解决各种的商业关键问题，如营销响应率、客户细分、客户忠诚度及终身价值、资产维护、资源规划、预测性维修、质量管理、社交媒体监测和情感分析等典型商业案例。

笔记

RapidMiner 解决方案覆盖了各个领域，包括汽车业、银行业、保险业、生命科学、制造业、石油和天然气、零售业及快消行业、通信业、以及公用事业等各个行业。RapidMiner 数据挖掘任务涉及范围广泛，包括各种数据艺术，能简化数据挖掘过程的设计和评价过程。它的特点是图形用户界面的互动原型，其功能和特点如下：

① 免费提供数据挖掘技术和库，100%用 Java 代码（可运行在操作系统）、Java API（应用编程接口），数据挖掘过程简单、强大和直观。

② 内部 XML 保证了标准化的格式来表示交换数据的挖掘过程。

③ 可以用简单脚本语言自动进行大规模进程，命令行（批处理模式）自动大规模应用。

④ 多层次的数据视图，确保有效和透明的数据。

⑤ 图形用户界面的互动原型，强大的可视化引擎，许多尖端的高维数据的可视化建模。

⑥ 简单的插件和推广机制。

⑦ 400 多个数据挖掘运营商的支持。

耶鲁大学已成功地将其应用在许多不同的应用领域，包括文本挖掘、多媒体挖掘、功能设计、数据流挖掘、集成开发的方法和分布式数据挖掘。

（6）Pentaho BI

Pentaho BI 商务智能平台不同于传统的商务智能产品（简称 BI），它是一个以流程为中心的，面向解决方案（Solution）的框架。其目的在于将一系列企业级 BI 产品、开源软件、API 等组件集成起来，方便商务智能应用的开发。它的出现，使得一系列面向商务智能的独立产品如 Jfree、Quartz 等，能够集成在一起，构成一项项复杂的、完整的商务智能解决方案，如图 1-10 所示。

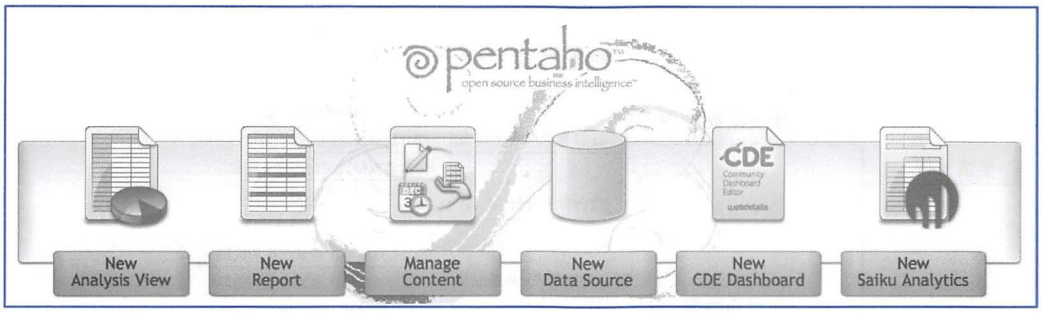



图 1-10　Pentaho BI 功能图

Pentaho BI 平台是 Pentaho Open BI 套件的核心架构和基础，是以流程为中心的，因为其中枢控制器是一个工作流引擎。工作流引擎使用流程定义来定义在 BI 平台上执行的商业智能流程。流程可以很容易被定制，也可以添加新的流程。BI 平台包含组件和报表，用以分析这些流程的性能。目前，Pentaho 的主要组成元素包括报表生成、分析、数据挖掘和工作流管理等。这些组件通过 J2EE、WebService、SOAP、HTTP、Java、JavaScript、Portals 等技术集成到 Pentaho 平台中。Pentaho 的发行，主要以 Pentaho SDK 的形式进行。

Pentaho SDK 共包含 Pentaho 平台、Pentaho 示例数据库、可独立运行的 Pentaho 平台、Pentaho 解决方案示例和一个预先配制好的 Pentaho 网络服务器 5 个部分。其中 Pentaho 平台是最主要的部分，囊括了 Pentaho 平台源代码的主体；Pentaho 数据库为 Pentaho 平台的正常运行提供的数据服务，包括配置信息、Solution 相关的信息等，对于 Pentaho 平台来说它不是必需的，通过配置是可以用其他数据库服务取代的；可独立运行的 Pentaho 平台是 Pentaho 平台的独立运行模式的示例，它演示了如何使 Pentaho 平台在没有应用服务器支持的情况下独立运行；Pentaho 解决方案示例是一个 Eclipse 工程，用来演示如何为 Pentaho 平台开发相关的商业智能解决方案。

## 任务实施

微课：1-2 学情分析系统设计

### 1. 案例构思

学情分析是伴随现代教学设计理论产生的。现代教学设计理论认为，认真研究学生的实际需要、能力水平、认知倾向以及社会对人才的需求分析，“为学习者设计教

学”，优化教学过程，可以更有效地达成教学目标，提高教学效率。为此，米老师需要设计一个学情分析系统，该学习系统不仅可以辅助教学，还能够通过对学生成绩、就业岗位与技能数据的分析，为学生推荐合适的工作岗位，及时补充技能知识。如图 1-11 所示为米老师设计的学情分析系统的目标实现图。

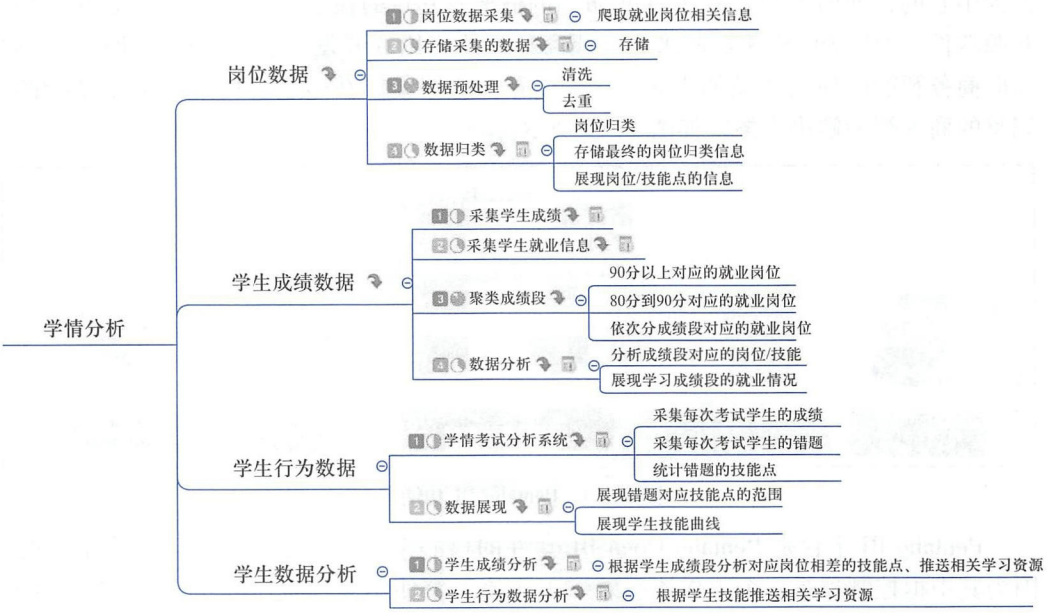



图 1-11　学情分析系统的目标实现图

## 2. 实现思路

（1）构建大数据平台

使用 Ambari 工具构建 Hadoop 大数据平台，该平台包括 HDFS 分布式文件系统、MapReduce 分布式计算框架、YARN 资源管理器、Hive 数据仓库、Mahout 数据挖掘工具、Sqoop 数据传输工具。

（2）采集岗位信息

借助 WebMagic 爬虫框架从招聘网站采集就业岗位数据，配置爬虫数据萃取表达式，对采集到的 HTML 内容提取出 CSV 格式，数据内容包括岗位、公司、月薪、福利、职位描述、职位要求等。对采集数据进行简单的筛选，保留有价值的信息。

（3）数据存储

HDFS 是一个分布式的文件系统，支持海量数据的存储，将采集的数据通过 Sqoop 传输工具上传到 HDFS 文件系统中。

（4）数据处理

MapReduce 是一个分布式的并行计算框架，具有高性价比和简单性的优点，MapReduce 模型可以将海量任务并行处理，使用该工具可以对数据进行清洗。Hive 数据仓库可以将 HiveSQL 转换成 MapReduce 任务进行处理，这里可以将数据放在 Hive 中

进行查询和过滤。

（5）数据分析

借助 Mahout 数据挖掘工具，使用算法库中的 K-means 文本聚类的方法对岗位信息数据进行分析挖掘，然后将分析结果与学生数据采用向量的相似度算法进行岗位匹配与推荐。

（6）数据可视化

采用电子表格（Excel）与前端开发工具（ECharts、D3. js）将分析结果直观地展现出来。

（7）学情分析系统工程运行

结合 Java Web 开发框架，将数据采集、数据清洗、数据分析以及数据可视化串联为一个系统，完成一套岗位分析+学情分析的产品 DEMO，界面主要用来展示数据可视化结果，设置相应的按钮进行数据计算服务，使用数据库中的数据源进行计算。

笔记

### 3. 系统整体架构

该学情分析案例数据库采用 MySQL 数据库，使用 Java Web SSH 开发框架（Spring+Struts+Hibernate）技术，采用 Tomcat 服务器作为部署环境。

在数据挖掘过程，采用了 WebMagic 网络爬虫来进行数据采集、MapReduce 计算框架来进行数据清洗、Mahout 来进行数据分析。

最后数据可视化层使用了 D3. js 和 Echarts 模板来进行前端页面开发。

Hadoop 平台使用的是 Ambari 平台，存储服务使用了 Hive 和 Hdfs。

学情分析系统架构图如图 1-12 所示。

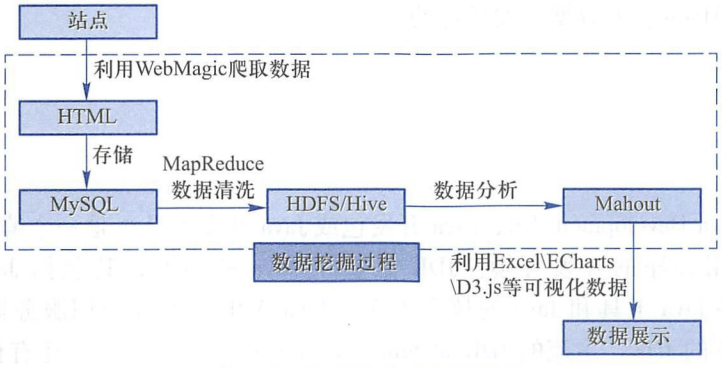



图 1-12　学情分析系统架构图

## 项目实训

**【实训任务】**

学情分析系统的设计。

**【实训目的】**

1. 了解大数据背景知识。

2. 熟悉大数据分析处理流程。
3. 领会学情分析系统架构设计。

【实训内容】

1. 查找合适的大数据解决方案。
2. 选择合适的学情系统分析工具 。
3. 设计学情分析系统。

## 任务 1.2 构建学情分析系统开发环境

### 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习 JDK、JRE 等相关 API，了解二者的差别。
2. 学习 Java 开发工具 Eclipse 集成开发工具的使用，以及相关配置。
3. 学习 Java Web 服务器 Tomcat 的安装和配置。
4. 学习 MySQL 服务器以及客户端的安装及配置。
5. 完成 Hadoop 大数据开发环境的配置。

### 任务目标

1. 学会 Java EE 开发工具的配置与使用。
2. 学会 MySQL 数据库和 Tomcat 的安装与配置。
3. 学会 Hadoop 大数据开发环境的配置。

### 知识准备

#### 1. JDK

JDK（Java Development Kit，Java 开发包或 Java 开发工具）是一个编写 Java Applet 小程序和应用程序的开发环境。JDK 是整个 Java 的核心，其包括 Java 运行环境（JRE）、一些 Java 工具和 Java 的核心类库（Java API），Java 应用服务器实质都是内置了某个版本的 JDK。主流的 JDK 是 Sun 公司发布的，除此之外，还有很多公司和组织都开发了自己的 JDK，例如，IBM 公司开发的 JDK，BEA 公司开发的 Jrocket，还有 GNU 组织开发的 JDK。

另外，可以把 Java API 类库中的 Java SE API 子集和 Java 虚拟机这两部分统称为 JRE，JRE 是支持 Java 程序运行的标准环境。

JRE 是运行环境，JDK 是开发环境。因此写 Java 程序的时候需要 JDK，而运行 Java 程序的时候就需要 JRE。而 JDK 里面已经包含了 JRE，因此只要安装了 JDK，就可以编辑和正常运行 Java 程序。但由于 JDK 包含了许多与运行无关的内容，占用的空间较大，因此运行普通的 Java 程序无须安装 JDK，而只需要安装 JRE 即可。

### 2. Java 开发工具 Eclipse

Eclipse 是一种可扩展的开放源代码的集成开发环境，是一款非常受欢迎的 Java 开发工具。Eclipse 的最大特点是接受各种开放源代码的插件，其中就包括了 JDT（Java Development Tools），利用它可以大大提高开发效率。

### 3. Tomcat Web 服务器

Tomcat 最初由 Sun 公司的软件构架师詹姆斯·邓肯·戴维森开发，后由 Sun 贡献给Apache 软件基金会，目前是 Apache 软件基金会的顶级项目。Tomcat 是一个免费且开源的 Web 服务器，支持 Servlet 和 JSP 规范。Tomcat 免费开源、技术先进、性能稳定，深受 Java 爱好者的喜爱并得到了部分软件开发商的认可，是目前比较流行的 Web 应用服务器。

### 4. MySQL 数据库

MySQL 是一个关系型数据库管理系统，最早由瑞典 MySQL AB 公司开发，目前为 Oracle 公司的产品。MySQL 是业内流行的关系型数据库，采用了双授权政策，分为社区版和商业版。由于其体积小、速度快、开放源码等特点，一般中小型网站的开发都选择 MySQL 作为数据库。

## 任务实施

微课：1-3 学情分析系统开发环境构建

大数据 Java SE 网络应用开发采用 Java 技术平台，在 Windows 7 下搭建开发环境需要安装 JDK、MySQL 数据库和 Eclipse。所有的安装软件都在发布的软件包内，无须下载。具体步骤如下：

（1）安装并配置 JDK

① 双击 JDK 安装程序，打开安装向导界面，如图 1-13 所示。

工具 1-1：JDK 安装包

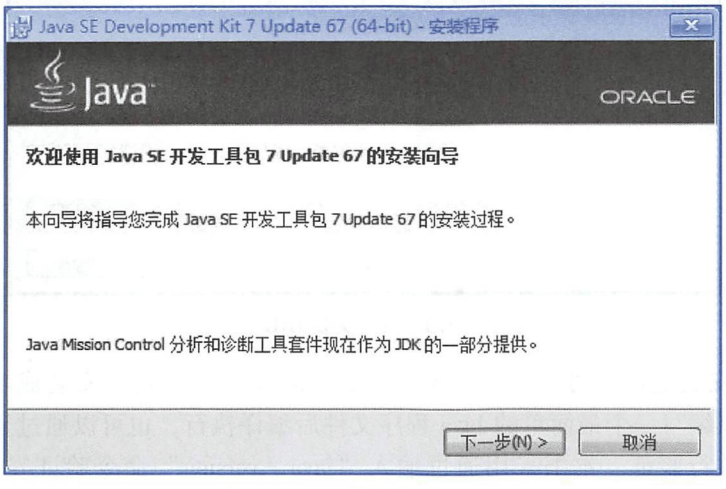



图 1-13　开始安装 Java JDK

② 设置安装路径，如图 1-14 所示。单击“更改”按钮，可以修改安装路径。完成后单击“下一步”按钮。

图 1-14 设置安装路径

③ 安装 JRE。在安装完 JDK 后，会弹出安装 JRE 窗口的界面（在这里要注意修改安装路径必须和 JDK 在同一目录下，而不是安装在 JDK 目录下），如图 1-15 所示。

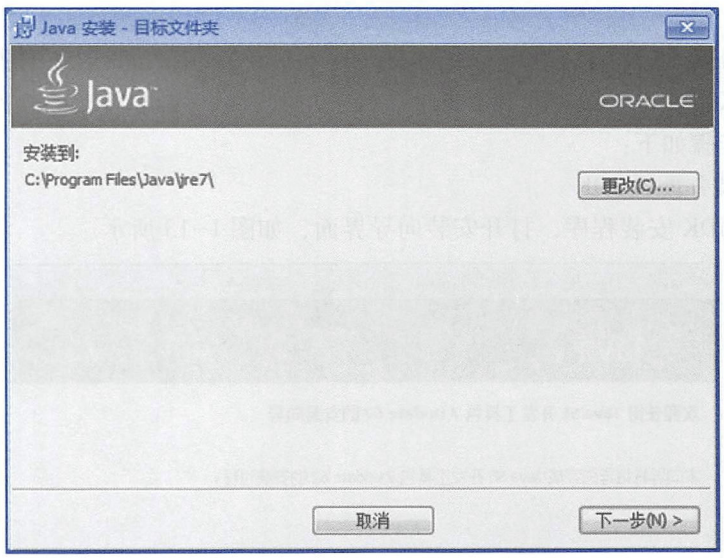



图 1-15 安装 JRE

④ 验证安装是否成功。安装完成后需要验证 Java 环境是否安装成功。验证方式有多种，可以编写一个最简单的 Java 程序文件后编译执行，也可以通过显示 Java 版本的命令方式进行验证。本书采用通过输入“java -version”命令验证安装是否成功。读者也可以通过该命令检查本机所安装的 Java 环境的版本，具体命令如图 1-16 所示。

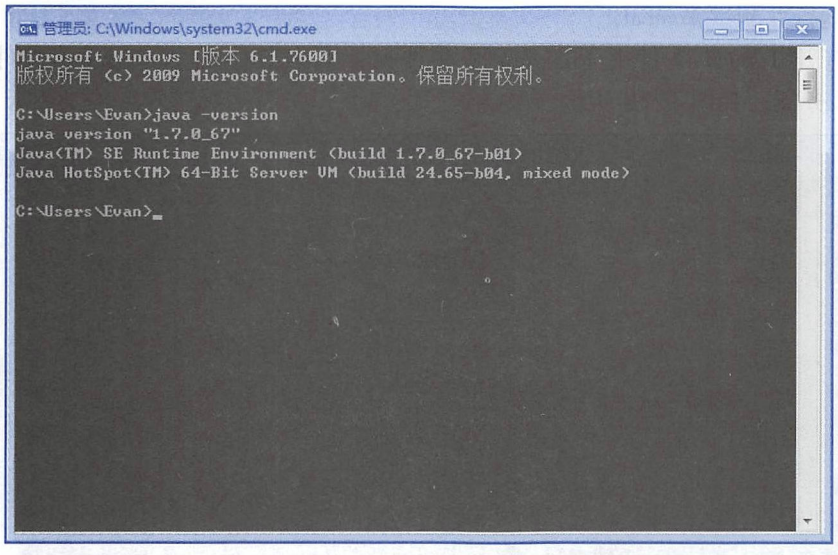



图 1-16　验证 Java 环境安装

（2）安装 Tomcat

1）下载并安装 Tomcat。

工具 1-2：Tomcat 安装包

① 从 Apache 网站（http://tomcat.apache.org/）下载 Tomcat，本书采用 Tomcat 7.0 版本开发。

② 将下载好的 Tomcat 压缩包解压至本地磁盘，如图 1-17 所示。

图 1-17　Tomcat 目录结构

2）运行并测试 Tomcat。

① 打开 Tomcat 文件夹下的 bin 子文件夹，双击其中的 startup. bat 文件，启动 Tomcat 服务器，如图 1-18 所示。

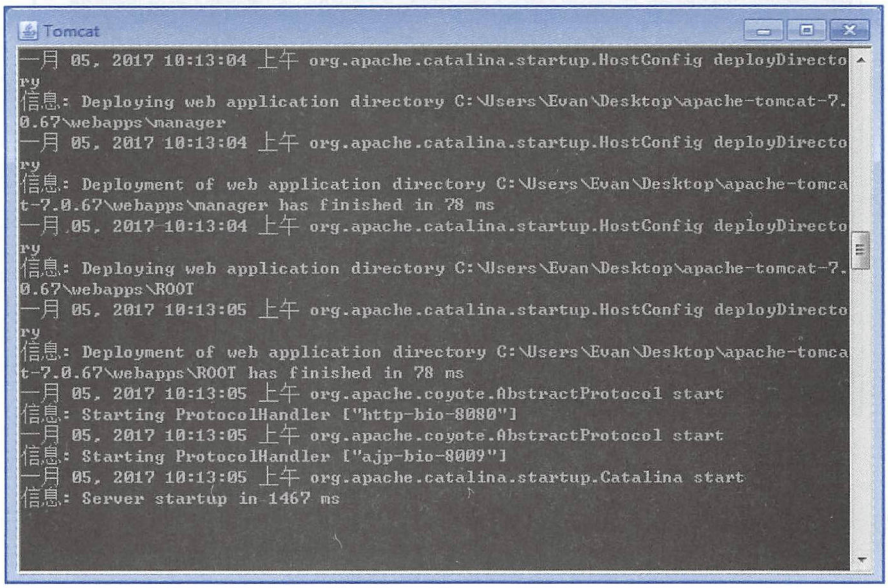



图 1-18 启动 Tomcat 服务器

② 在浏览器的地址栏内输入“http://localhost:8080”，如果 Tomcat 服务器部署成功，则显示如图 1-19 所示的界面。

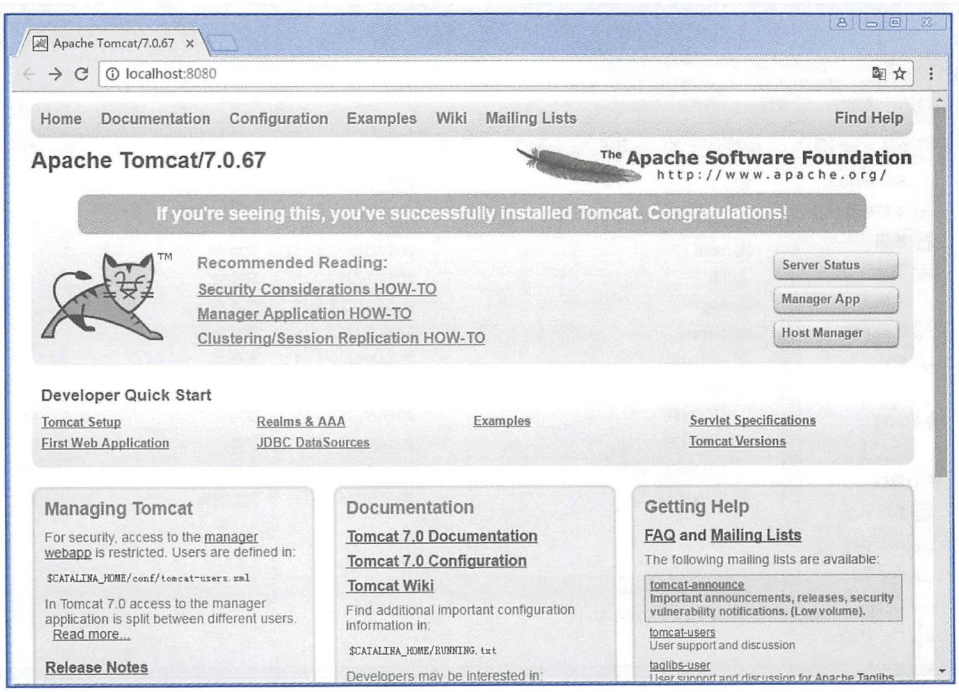



图 1-19 Tomcat 运行界面

③ 测试成功以后，关闭 Tomcat。

（3）安装并配置 Eclipse

工具 1-3：Eclipse 安装包

1）下载并安装 Eclipse。

① 从 Eclipse 的官网（http：//www. eclipse. org）网站获取到最新版本的 Eclipse，一般选择下载 Eclipse IDE for Java EE Developers 版本。

② 将下载好的 Eclipse 压缩包解压至本地磁盘，解压好的文件如图 1-20 所示。

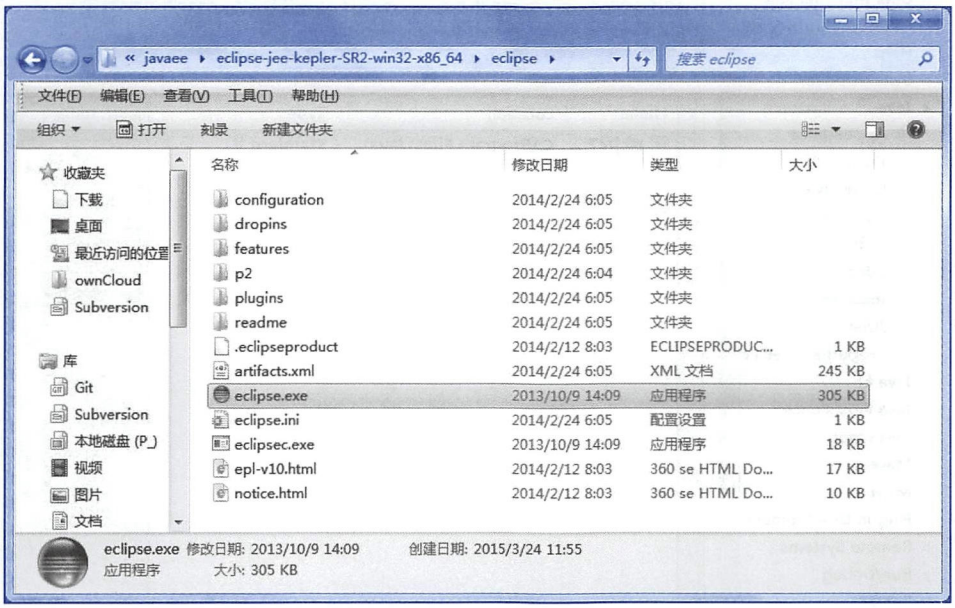



图 1-20　Eclipse 目录结构

2）运行并配置 Eclipse。

① 双击 eclipse. exe 文件启动 Eclipse，单击“Browse”按钮，在打开的对话框中选择工作目录，注意该目录不能有中文字符，完成后单击“OK”按钮，如图 1-21 所示。

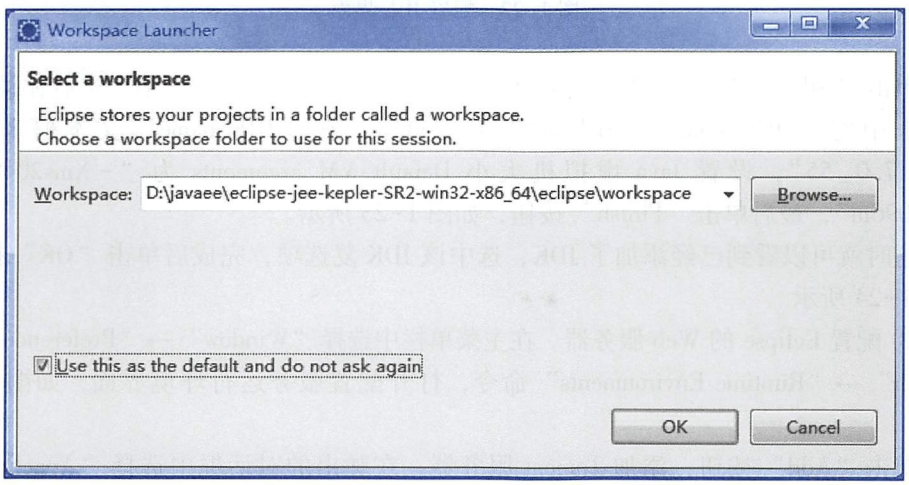



图 1-21　Eclipse 选择工作目录

② 配置 Eclipse 的 JDK 环境。

在主菜单栏中选择“Window”→“Preferences”→“Java”→“Installed JREs”命令，打开配置 JRE 界面，如图 1-22 所示。

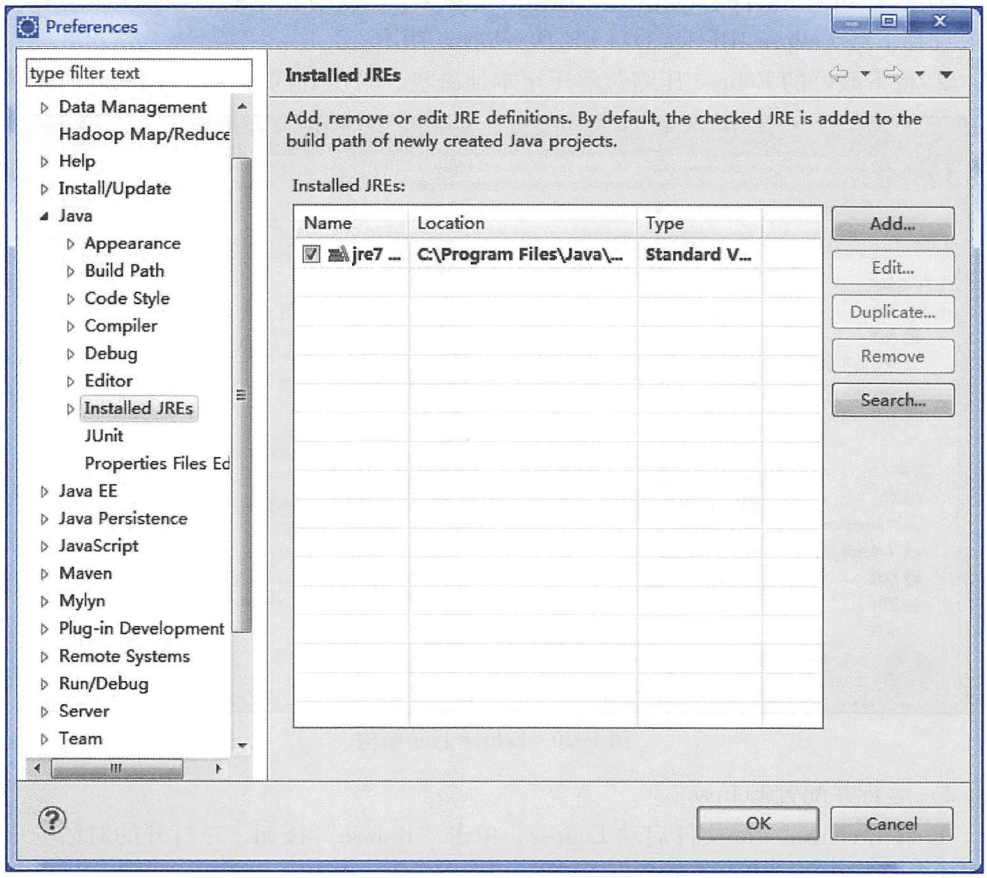



图 1-22 配置 JRE 界面

单击“Add”按钮，在弹出的对话框中选中“Standard VM”选项。然后在跳转的对话框中配置 JRE home 为 JDK 的安装主目录，在“JRE name”文本框中输入“jdk1.7.0_55”，设置 Java 虚拟机大小 Default VM arguments 为“-Xms2048m -Xmx4096m”，最后单击“Finish”按钮，如图 1-23 所示。

这时就可以看到已经添加了 JDK，选中该 JDK 复选项，完成后单击“OK”按钮，如图 1-24 所示。

③ 配置 Eclipse 的 Web 服务器。在主菜单栏中选择“Window”→“Preferences”→“Server”→“Runtime Environments”命令，打开配置服务运行环境界面，如图 1-25 所示。

单击“Add”按钮，添加 Tomcat 服务器。在弹出的对话框中选择“Apache”→“Apache Tomcat v7.0”选项，单击“Next”按钮，如图 1-26 所示。

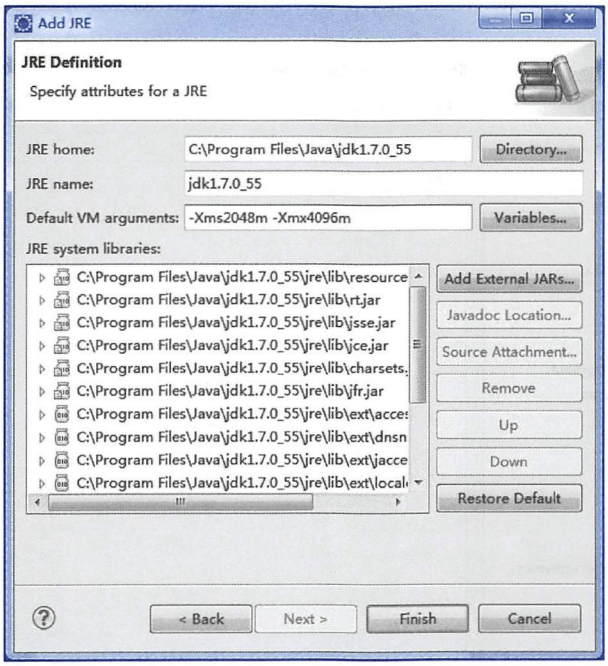



图 1-23　配置 JDK 路径

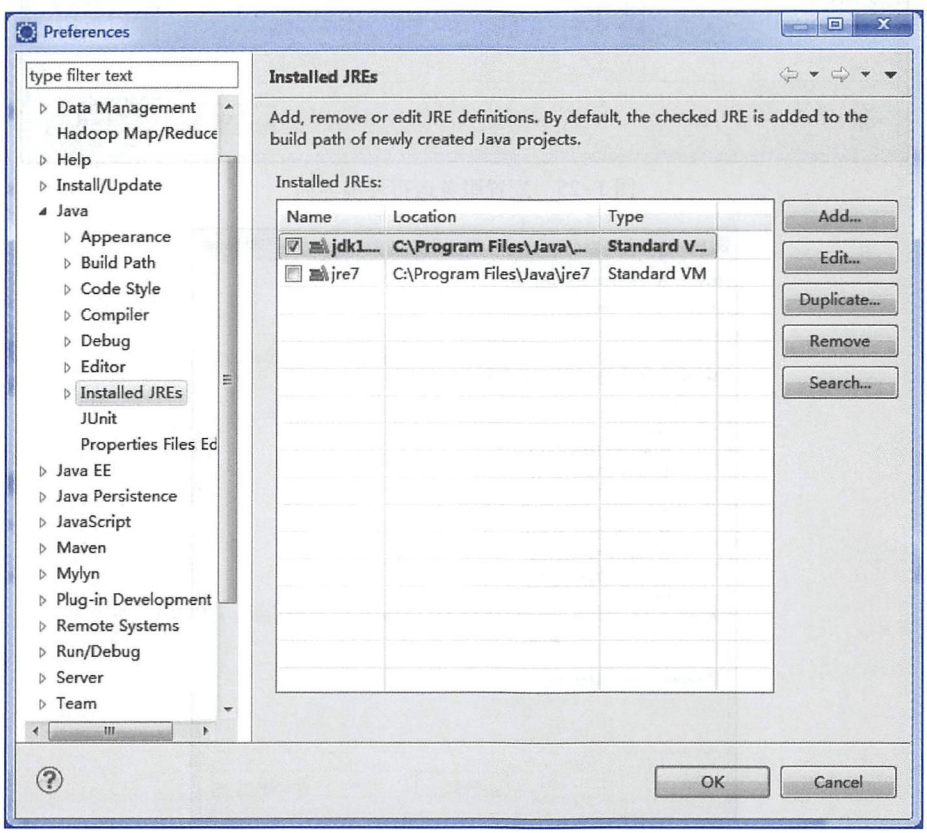



图 1-24　选择 JDK

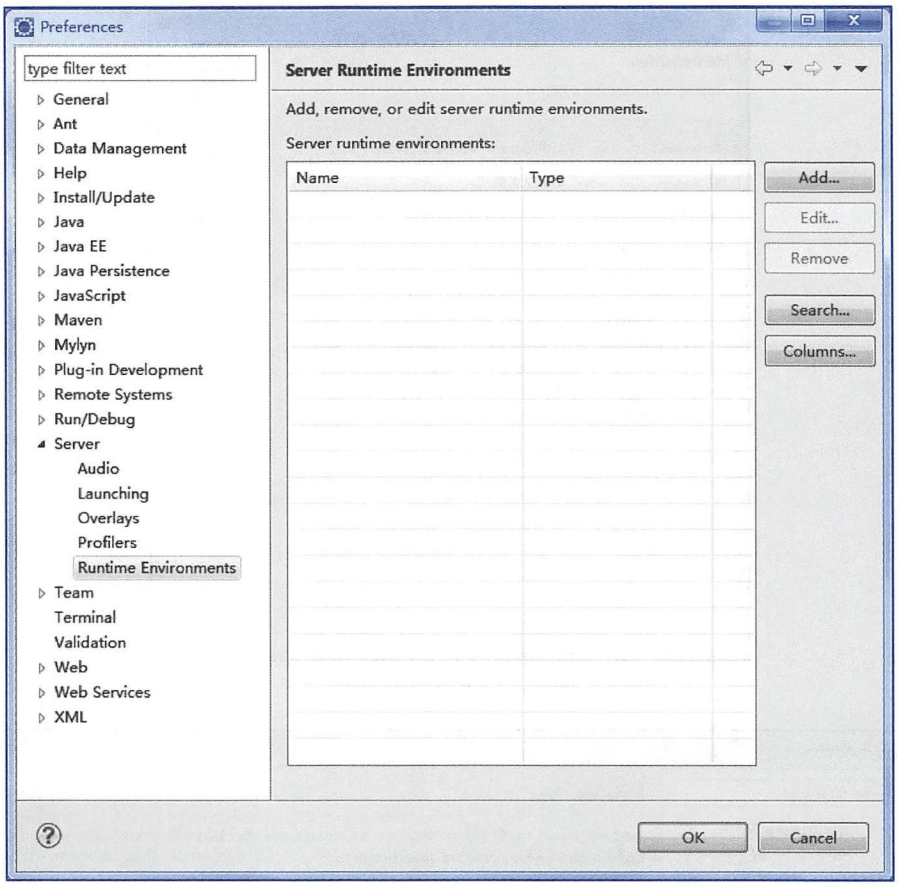



图 1-25 配置服务运行环境界面

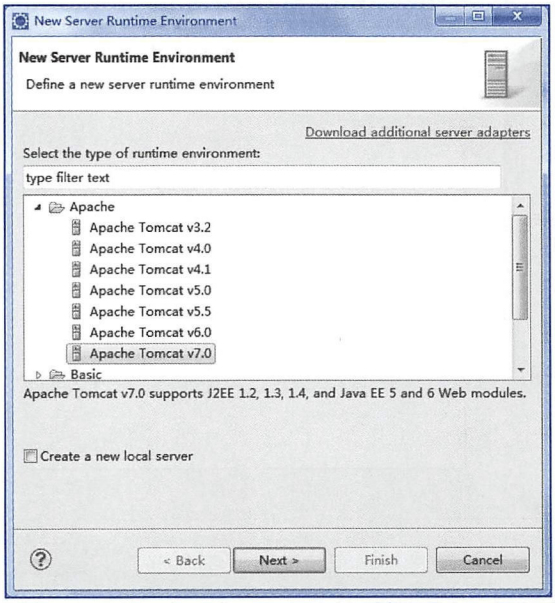



图 1-26 选择 Tomcat 服务器

设置 Tomcat 安装路径（Tomcat installation directory）和 JRE，完成后单击“Finish”按钮，如图 1-27 所示。

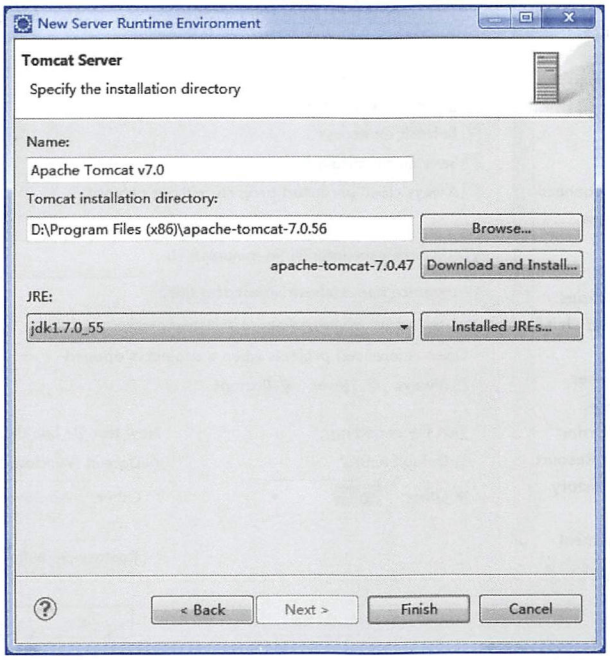



图 1-27　配置 Tomcat

在主菜单栏中选择“Window”→“Show View”→“Servers”命令，可以打开 Servers 窗口，添加刚才创建的 Tomcat 服务器，在此窗口中可以直接停止或者启动调试 Tomcat 服务器，如图 1-28 所示。

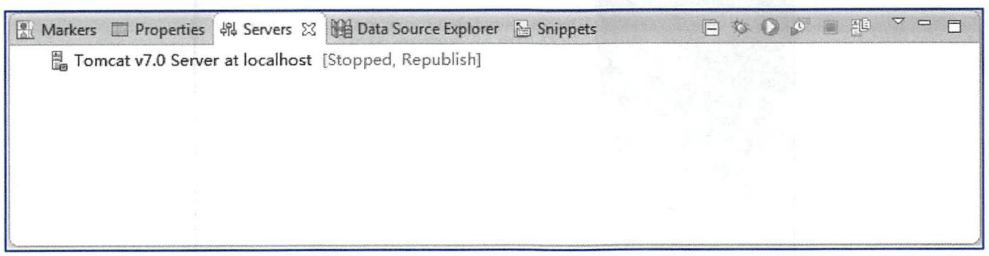



图 1-28　添加 Tomcat 服务器

④ 设置 Eclipse 编码方式。在主菜单栏中选择“Window”→“Preferences”→“General”→“Workspace”命令，在“Text file encoding”项目栏中设置工作空间的代码编码方式为 UTF-8，完成后单击“OK”按钮，如图 1-29 所示。

至此 Eclipse IDE 安装完成。

（4）安装并配置 MySQL Server

① 双击打开 MySQL 5.0 安装程序 MySQL\Setup.exe，在打开的 MySQL 安装向导对话框中单击“Next”按钮，如图 1-30 所示。

工具 1-4：MySQL 5.0 安装包

版权所有，严禁传播，违者自负法律责任！



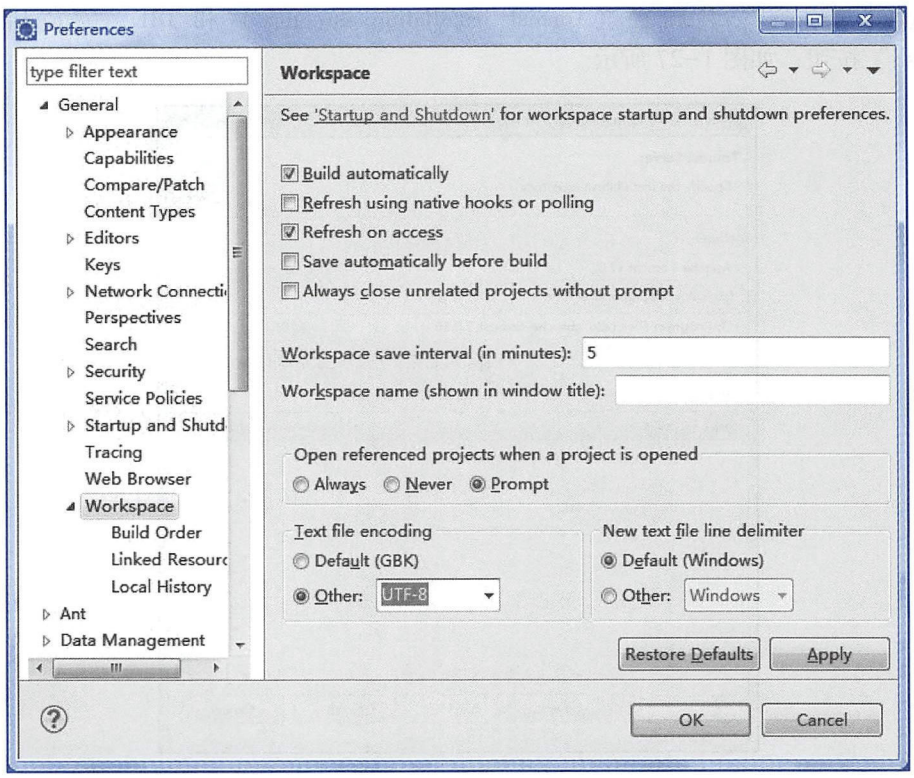



图 1-29 设置程序编码方式

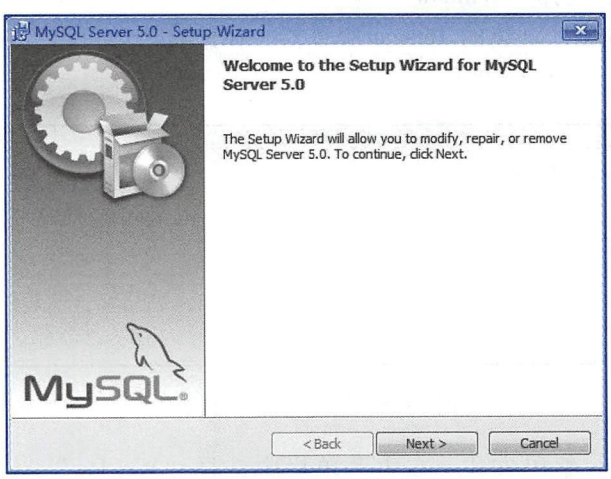



图 1-30 MySQL 安装向导

② 在选择安装类型界面中，有 Typical（默认）、Complete（完全）、Custom（用户自定义）3 个选项，选中“Custom”单选按钮，单击“Next”按钮，如图 1-31 所示。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



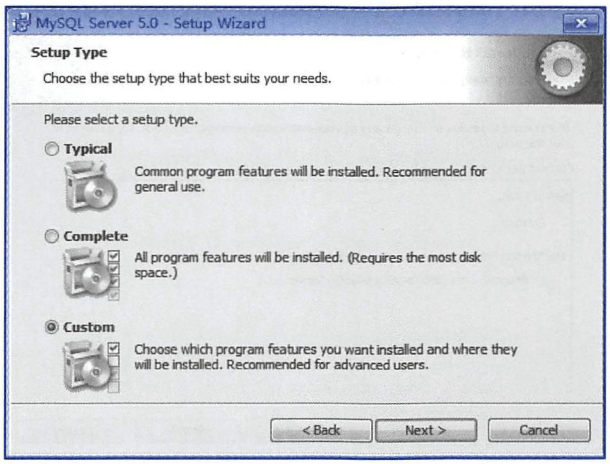



图 1-31　选择安装的类型

③ 在自定义安装界面中选择 MySQL 数据库的安装路径，本项目设置的路径是 D:\Program File\MySQL。单击 Developer Components（开发者部分）项前的下拉按钮▼，在弹出的下拉列表中选择“This feature, and all subfeatures, will be installed on local hard drive.”选项。完成上述操作后单击“Next”按钮，如图 1-32 所示。

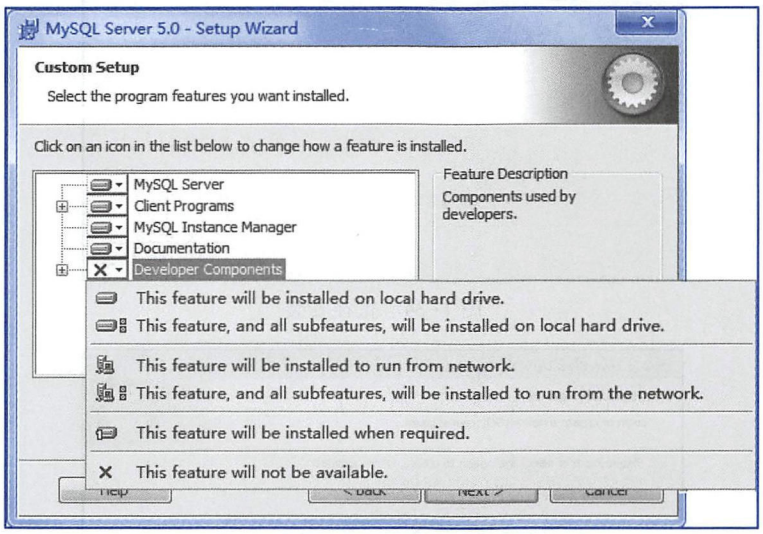



图 1-32　安装设置

④ 打开准备安装界面，首先确认刚才的设置是否正确，如果有误，单击“Back”按钮返回，可以重新设置。如果之前的配置没有错误，单击“Install”按钮继续安装，如图 1-33 所示。

⑤ 打开正在安装界面，如图 1-34 所示。

⑥ 安装完成后出现注册界面，选中“Skip Sign-Up”单选按钮，单击“Next”按钮，如图 1-35 所示。

非卖品，仅供非商业用途或交流学习使用　　严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



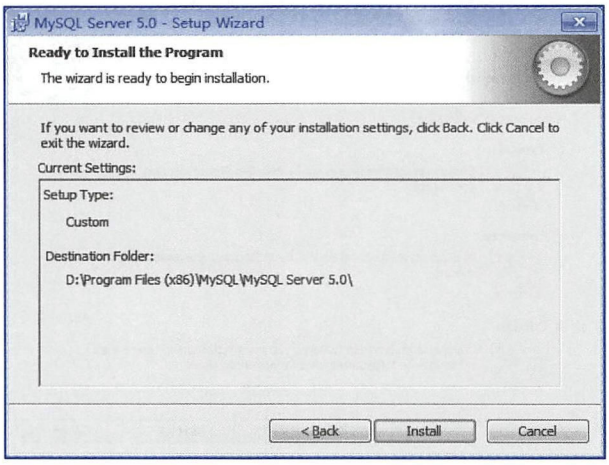



图 1-33 准备安装

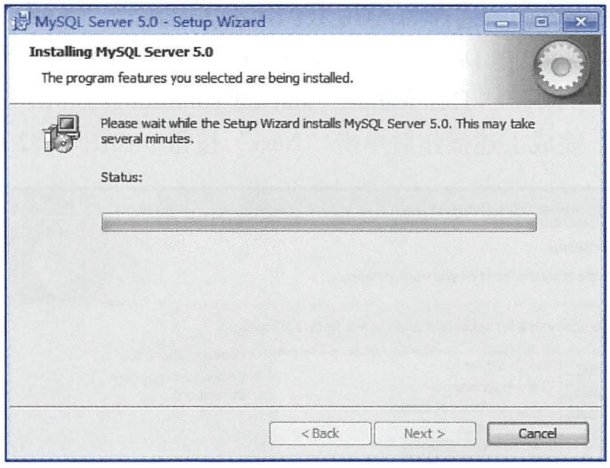



图 1-34 正在安装

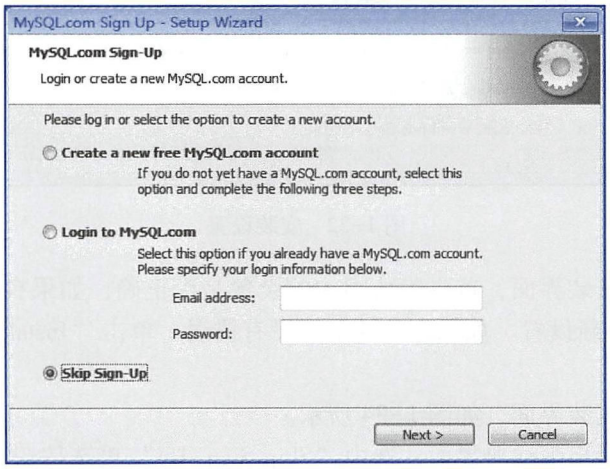



图 1-35 注册界面

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



⑦ 单击“Finish”按钮，完成 MySQL 安装，如图 1-36 所示。

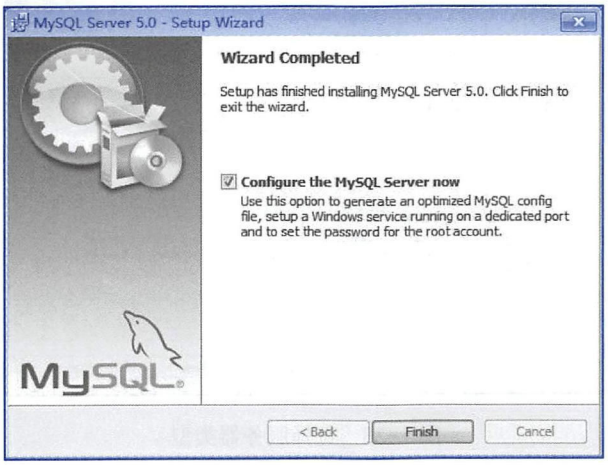



图 1-36　MySQL 安装完成

⑧ MySQL 数据库安装完成之后，出现如图 1-37 所示的配置向导，单击“Next”按钮。

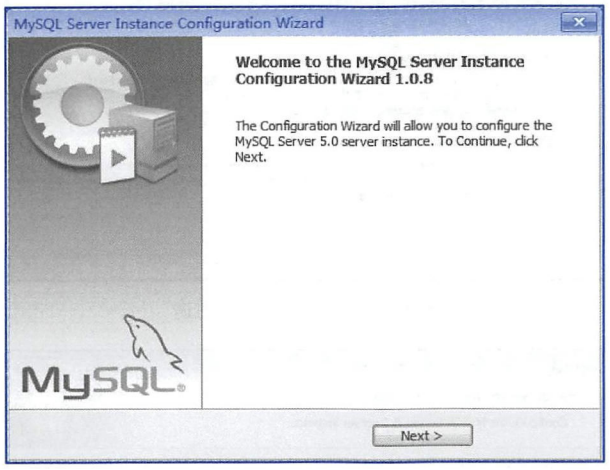



图 1-37　MySQL 配置向导

⑨ 在配置方式中选中“Detailed Configuration”（详细配置）单选按钮，单击“Next”按钮，如图 1-38 所示。

⑩ 设置 MySQL 数据库的用途，选中“Multifunctional Database”（通用多功能型）单选按钮，单击“Next”按钮，如图 1-39 所示。

⑪ 对 InnoDB Tablespace 进行配置，选择 InnoDB 数据库文件存储位置。单击 Installation Path（应用程序安装路径）旁的下拉按钮▼，选择默认安装目录，单击“Next”按钮，如图 1-40 所示。

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



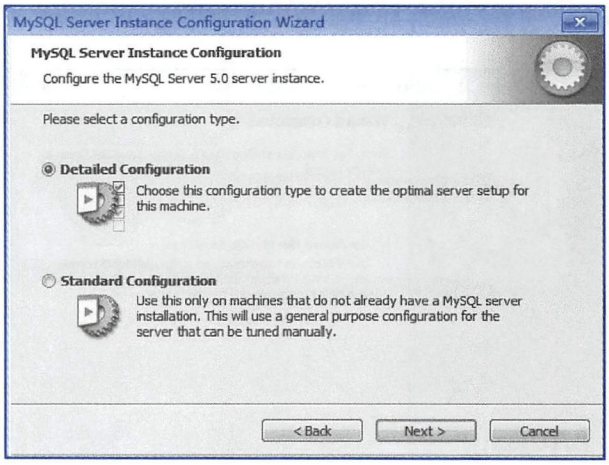



图 1-38 选择服务器类型

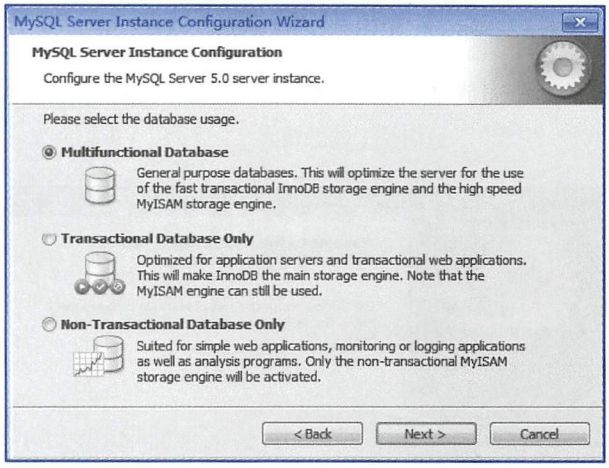



图 1-39 选择数据库用途

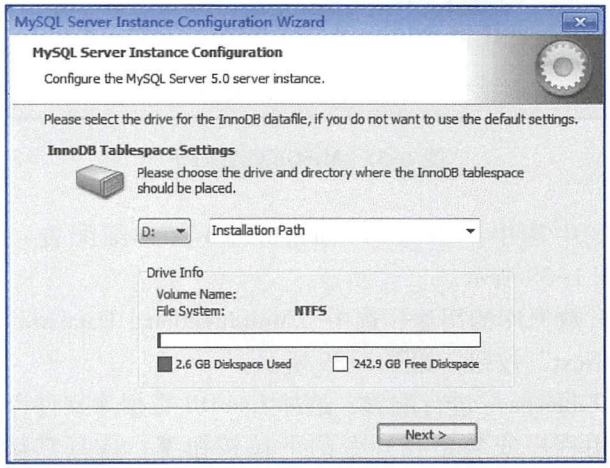



图 1-40 选择存储位置

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



⑫ 配置 MySQL 服务器的参数，选中“Manual Setting”（手动设置）单选按钮，单击“Next”按钮，如图 1-41 所示。

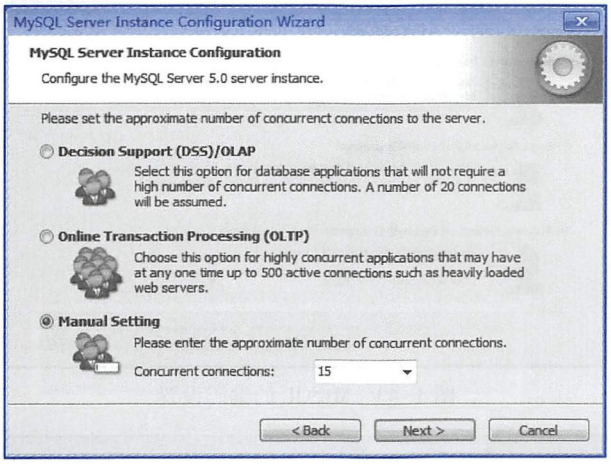



图 1-41　MySQL 并发参数的设置

⑬ 设置 TCP/IP 参数，选中“Enable TCP/IP Networking”单选按钮，默认的端口为 3306，选中“Enable Strict Mode”复选框，最后单击“Next”按钮，如图 1-42 所示。

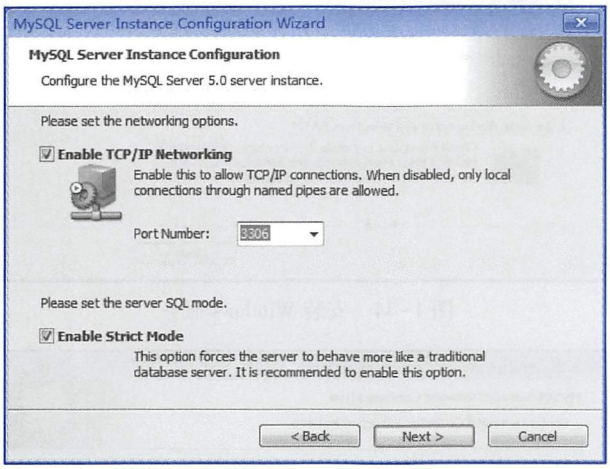



图 1-42　MySQL TCP 参数设置

⑭ 配置字符集配置，选中“Manual Selected Default Character Set/Collation”单选按钮，在“Character Set”下拉列表中选择“utf8”选项，单击“Next”按钮，如图 1-43 所示。

⑮ 将 MySQL 安装为 Windows 服务，选中“Install As Windows Service”和“Include Bin Direction In Windows PATH”复选框，单击“Next”按钮，如图 1-44 所示。

⑯ 设置管理员密码，管理员账号为 root。选中“Modify Security Settings”和“Enable root access from remote machines”复选框，在“New root password”文本框中输入密码，并在“Confirm”文本框中再次输入该密码，最后单击“Next”按钮，如图 1-45 所示。

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！

图 1-43 MySQL 的编码设置

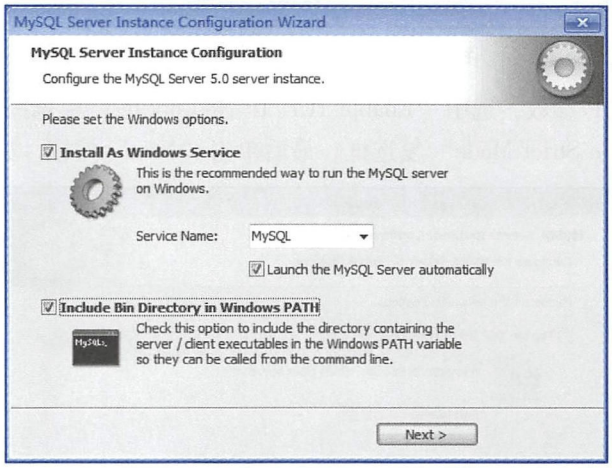



图 1-44 安装 Windows 服务

图 1-45 登录密码参数设置

非卖品，仅供非商业用途或交流学习使用

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



⑰ 经过以上操作，MySQL 的所有配置都已设置完成，单击“Execute”按钮执行配置，如图 1-46 所示。

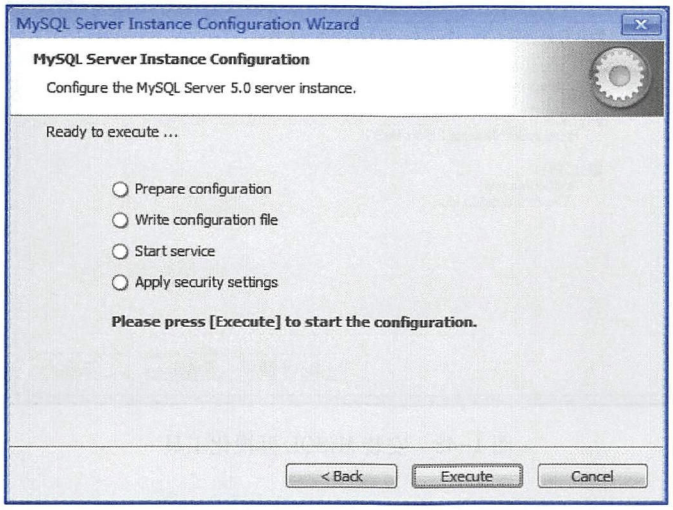



图 1-46　执行配置

⑱ 安装成功后，单击“Finish”按钮，如图 1-47 所示。

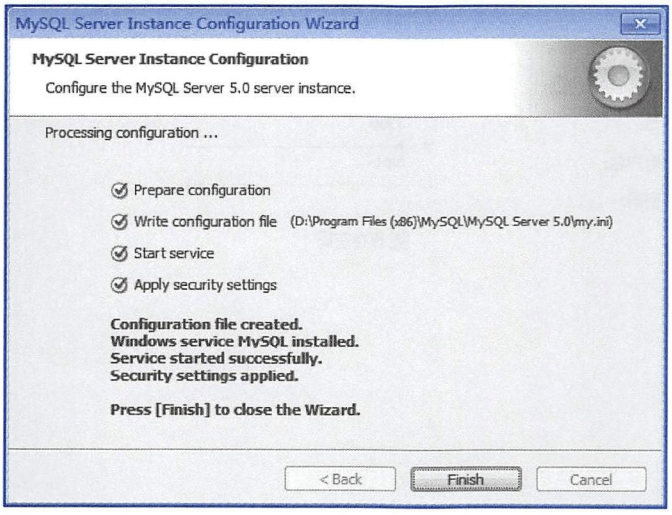



图 1-47　安装成功

（5）安装 MySQL 客户端

① 使用 Navicat 作为 MySQL 的客户端管理工具，运行 Navicat 8.0.exe 安装程序，设置安装目录后，单击“安装”按钮，如图 1-48 所示。

工具 1-5：Navicat 8.0 安装包

② 打开 Navicat，测试与本地 MySQL 数据库连接，输入主机名、埠（端口号）、

非卖品，仅供非商业用途或交流学习使用　　严禁网络传播本PDF，违者责任自负！

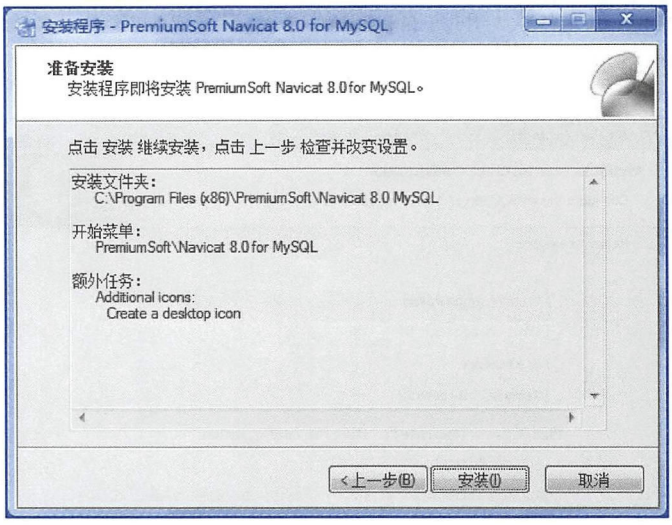



图 1-48 安装 MySQL 可视化工具

用户名与密码参数，单击“确定”按钮，如图 1-49 所示。

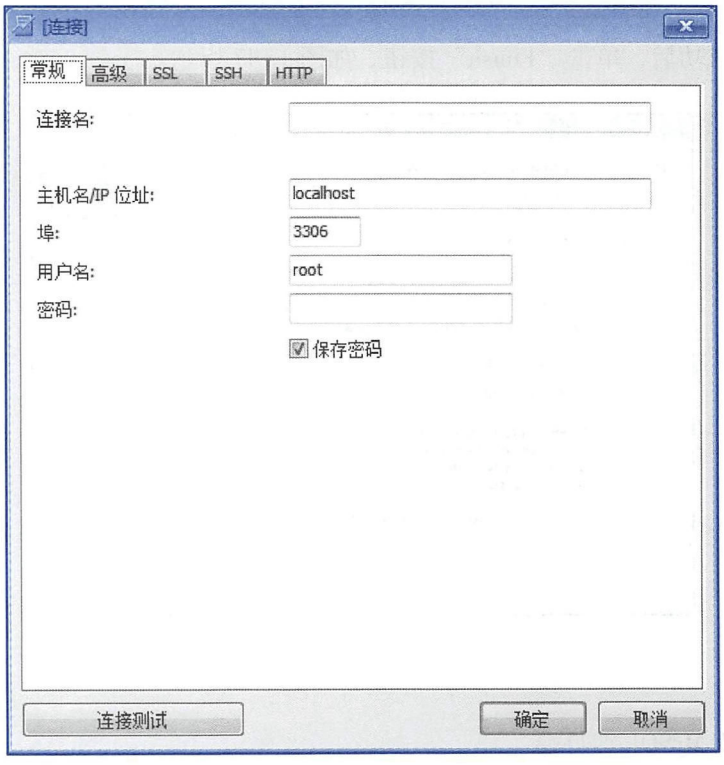



图 1-49 输入数据库连接参数

③ 单击“连接测试”按钮，提示连接成功，如图 1-50 所示。

严禁网络传播本PDF，违者责任自负！

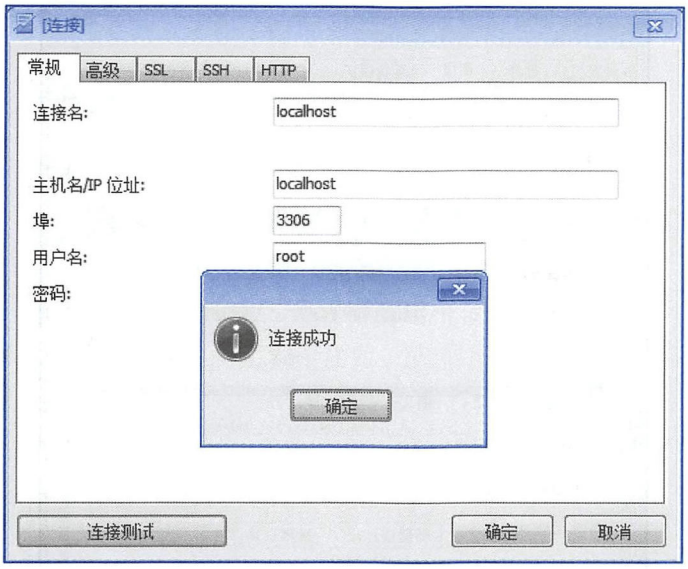



图 1-50　测试连接是否成功

④ 进入到 Navicat 主界面，用户可以对指定的数据库进行操作和管理，如图 1-51 所示。

图 1-51　Navicat 主界面

（6）构建 Hadoop 开发平台

运行案例前，需要对 Hadoop 的开发环境进行配置。

工具 1-6：Hadoop 安装包

首先在用于开发的个人计算机上安装 Hadoop 2.7.0，软件包为 hadoop-2.7.0.zip，本项目中解压到 D:\Program Files（x86）路径中。将 hadoop-2.7.0\bin 目录下的 hadoop.dll 文件复制到系统路径 C:\Windows\System32 下。替换复制完成后，将 Hadoop 安装目录添加到系统环境变量中，在用户变量中添加名为 HADOOP_HOME 变量（变量值为 hadoop-2.7.0 的解压路径），如图 1-52 所示。

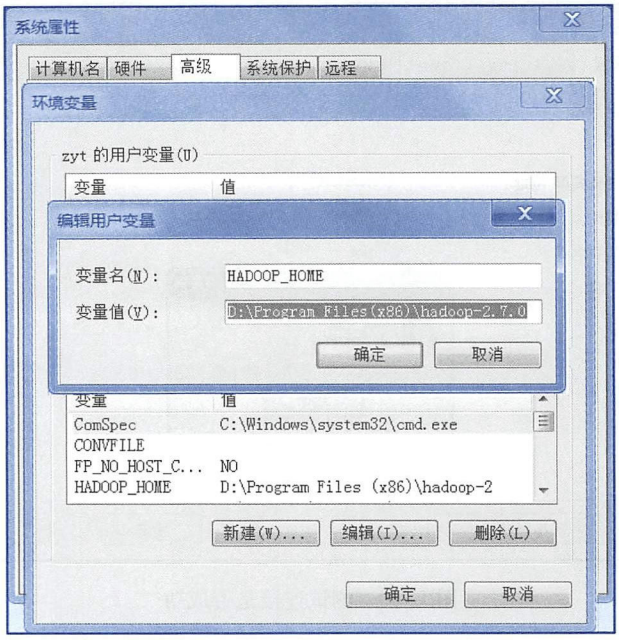



图 1-52 Hadoop 配置环境变量

之后在系统变量的 Path 中添加 Hadoop 的 bin 目录路径，如图 1-53 所示。

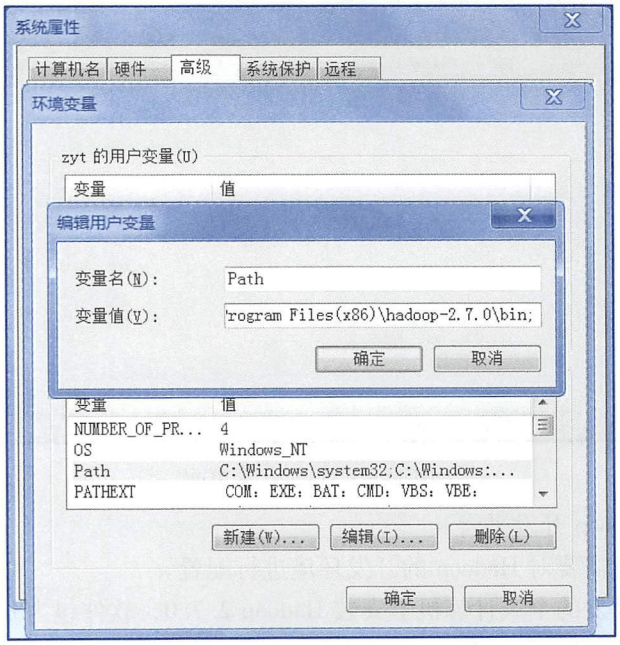



图 1-53 Hadoop 配置环境变量 Path

工具 1-7：Eclipse 插件

添加完成后，将关于 Hadoop 的 Eclipse 插件 hadoop-eclipse-kepler-plugin-2.2.0.jar 文件复制到 Eclipse 目录下的 Plugins 目录中，如图 1-54 所示。

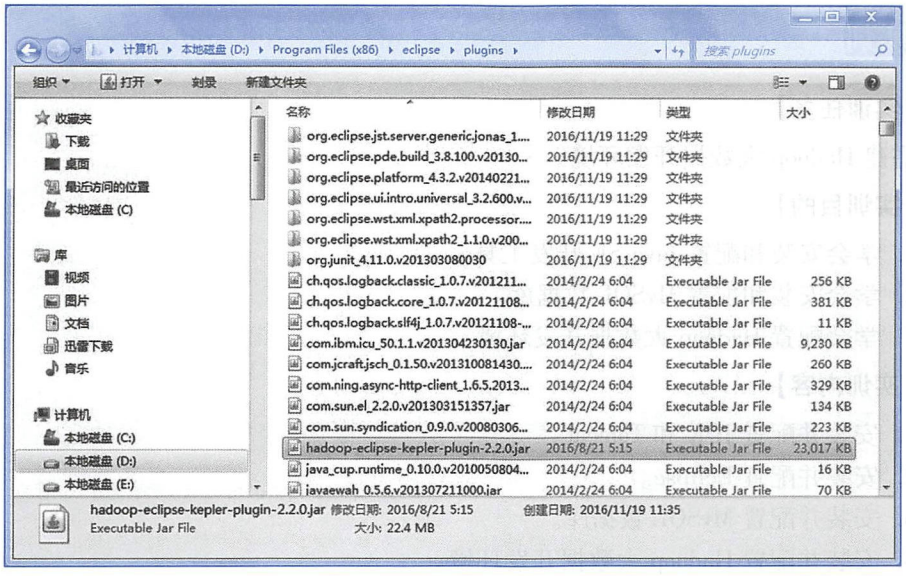



图 1-54　配置 Hadoop 的 Eclipse 插件

插件复制完成后，重新启动 Eclipse，配置 Hadoop 插件。在主菜单栏中选择“Window”→“Preferences”→“Hadoop Map/Reduce”命令，在打开的对话框“Hadoop installation diretory”文本框中输入 hadoop-2.7.0 的目录路径，最后单击“OK”按钮，如图 1-55 所示。

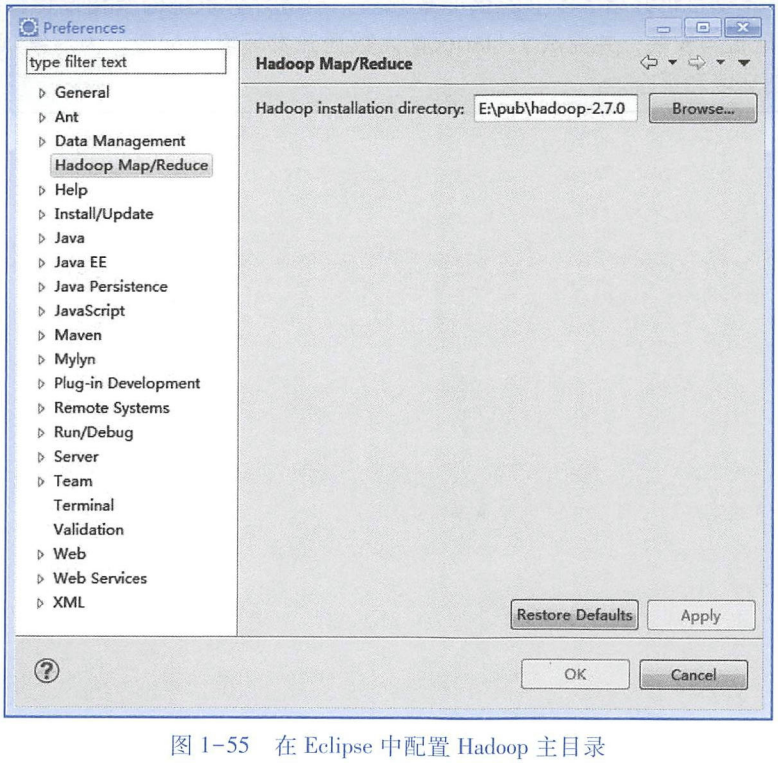



图 1-55　在 Eclipse 中配置 Hadoop 主目录

## 项目实训

**【实训任务】**

搭建 Hadoop 大数据开发环境。

**【实训目的】**

1. 学会安装和配置 Java SE 开发工具。
2. 学会安装和配置 MySQL 数据库。
3. 学会配置 Hadoop 大数据开发环境。

**【实训内容】**

1. 安装并配置 JDK 和 Tomcat。
2. 安装并配置 Eclipse。
3. 安装并配置 MySQL 数据库。
4. 安装并配置 Hadoop 大数据开发环境。

## 项目小结

本项目全面阐释了大数据的概念、相关的技术和应用的现状，使读者对大数据的相关知识、产业链以及大数据分析流程、工具等有了清晰的认识，了解使用大数据的意义，为建立大数据思维奠定了良好的基础。

同时，本项目也介绍了本书的核心案例——学情分析系统的架构设计，包括 Java SE 的环境配置与安装，并完成了 Hadoop 大数据开发环境的配置，为后续项目的学习搭建了基础环境和平台。

# 项目 2

# Hadoop大数据平台的构建

PPT2：Hadoop 大数据平台的构建

## 学习目标

【知识目标】

- 掌握 Hadoop 大数据平台的基本内容。
- 了解 Hadoop 各组件的功能与联系。

【技能目标】

- 学会 Hadoop 大数据平台的搭建。
- 学会使用 Ambari 管理大数据平台。

## 学习情境

某公司预研发“就业和学情分析系统”，为职业教育提供大数据服务，实时跟踪学生学情数据，指导学生就业定向和职业规划。公司同时向学生推荐学习资源，制定科学合理的学习规划。公司联合某院校专家米老师，双方建立了项目合作。米老师是信息类专业的一线教学专家，负责从专业业务角度为项目研发组提供业务分析指导，公司技术人员参与开发并负责系统实现。

米老师通过分析学生专业课程技能考核、学生就业、学生学习兴趣、企业岗位需求、互联网招聘等数据信息，建立岗位模型、知识模型和学情模型，指导公司进行就业和学情分析系统的研发。

米老师在综合分析相关技术和应用实践的前提下，选择了主流的 Hadoop 大数据平台，并采用 Apache Ambari 管理平台构建 Hadoop 开发环境。

# 任务 2.1 搭建 Ambari Hadoop 系统

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习 Cloudera Manager、Ambari 和 MapR 等大数据平台管理工具。
2. 学习 Hadoop 生态系统的相关技术，了解各项技术的功能。
3. 完成 Ambari 的安装与配置。

## 任务目标

1. 了解 Hadoop 背景知识，熟悉 Hadoop 的生态系统。
2. 学会 Hadoop 分布式集群的组建方法。
3. 学会 Ambari Server 和 Ambari Agent 的配置方法。

## 知识准备

### 1. Hadoop 平台产品

Hadoop 是 Apache Lucene 创始人 Doug Cutting 创建的，Lucene 是一个广泛使用的文本搜索系统库。Hadoop 起源于一个开源的网络搜索引擎 Apache Nutch，它本身也是 Lucene 项目的一部分。

笔记

Nutch 项目始于 2002 年，这是一个可以运行的网页爬取工具和搜索引擎系统。后来，其开发者认为这一架构可扩展度不够，不能解决数十亿网页的搜索问题。但是，随后出现的谷歌分布式文件系统 GFS 可以解决在网页爬取和索引过程中产生的较大文件的存储需求，并能够节省系统管理所花费的大量时间。因此，在 2004 年，他们开始着手实现开源的分布式文件系统 NDFS。

2004 年，谷歌公司发布了 MapReduce。2005 年，Nutch 的开发者在 Nutch 上发布了一个可工作的 MapReduce 应用，并将所有主要的 Nutch 算法移植到使用 MapReduce 的 NDFS 来运行。

Nutch 中的 NDFS 及其实现的应用远不只是搜索领域。2006 年 2 月，从 Nutch 中转移出一个独立的 Lucene 子项目，成为 Hadoop。大约在同一时间，Doug Cutting 加入雅虎公司（Yahoo）。雅虎公司提供专门的团队和资源将 Hadoop 发展成一个可在网络上运行的系统。2008 年 2 月，雅虎公司宣布其搜索引擎产品部署在一个拥有 1 万个内核的 Hadoop 集群上。

互联网巨头雅虎公司作为 Hadoop 框架的先驱研究者，已经将 Hadoop 打造成卓越的大数据平台。但相比于 SQL 数据库，Hadoop 在某些方面仍然显得不够完善。这直接导致现今所有目光都集中在 Hadoop 供应商上，其中 Cloudera、Hortonworks、MapR

是最具有代表性的公司，它们带来众多的创新并提供强大的集群工具。这些管理工具分别是 Cloudera Manager、Hortonworks 和 MapR，它们均能对集群进行自动化安装、管理、监控、诊断和集成。如图 2-1 所示为 3 大 Hadoop 平台的供应商，以下是这些公司的产品服务。

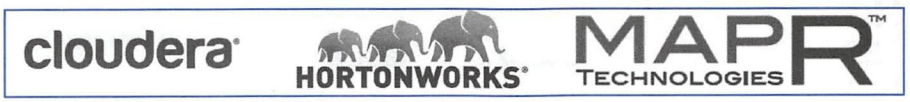



图 2-1　三大 Hadoop 平台的供应商

（1）Cloudera

Cloudera 的发行版本 CDH 和 Hortonworks 的发行版本 HDP，均是基于 Apache Hadoop 衍生出来的，并且免费提供，它们的收费项绝大多数在于服务维护。对于国内大多数的企业，它们的数据平台绝大多数选择 CDH、HDP 或者 MapR，主要是因为这些公司提供的 Hadoop 产品版本的划分非常清晰。例如，Cloudera 提供的 CDH3 和 CDH4，分别对应第一代 Hadoop（Hadoop 1.0）和第二代 Hadoop（Hadoop 2.0），目前已经更新到 CDH5。相对而言，Apache 版本则混乱得多。HDP 版本是比较新的版本，目前与 Apache 基本同步，因为 Hortonworks 内部大部分员工都是 Apache 代码贡献者，尤其是 Hadoop 2.0 的贡献者。

关于 Cloudera 的 CDH 和其他相关开放源代码项目，其不同之处在于，它包括了 Impala 和 Cloudera Search。Cloudera Impala 是一种 MPP（大规模并行处理）SQL 引擎，用于交互式分析和商业智能。它可以查询来自各种源的 Hadoop 数据文件，包括由 MapReduce 作业生成的数据文件或加载到 Hive 表中的数据文件。YARN 资源管理组件让 Impala 能够共存于使用 Impala SQL 查询并发运行批处理工作负载的群集上。可以通过 Cloudera Manager 用户界面管理 Impala 及其他 Hadoop 组件，并通过 Sentry 授权框架保护其数据。

Cloudera Search 主要提供近实时（Near-Real-Time）访问已存储的数据或者摄取数据到 Hadoop 以及 HBase 中去。Search 提供了近实时的索引、批量索引、全文检索和 Drill-Down（下钻）的导航，以及一个简单的、全文检索的接口，需要一些 NoSQL 或者编程基础（技能）。Search 使用 CDH 中灵活的、可扩展的以及可靠的存储系统，这样不再需要在基础设施层或者业务层移动大量的数据，也不需要产生新的任务。

CDH 平台还有一款端到端数据管理工具 Cloudera Navigator，它使管理员、数据经理和分析师能够了解 Hadoop 中的大量数据，且简化了存储和密钥的管理。

Cloudera Manager 是一款在 Web 界面中部署、管理、监控 Hadoop 平台的工具，使管理集群简单而直接。它还包括 Cloudera Manager API，可用来获取群集运行状况信息、度量信息以及对 Cloudera Manager 进行配置，其管理界面如图 2-2 所示。

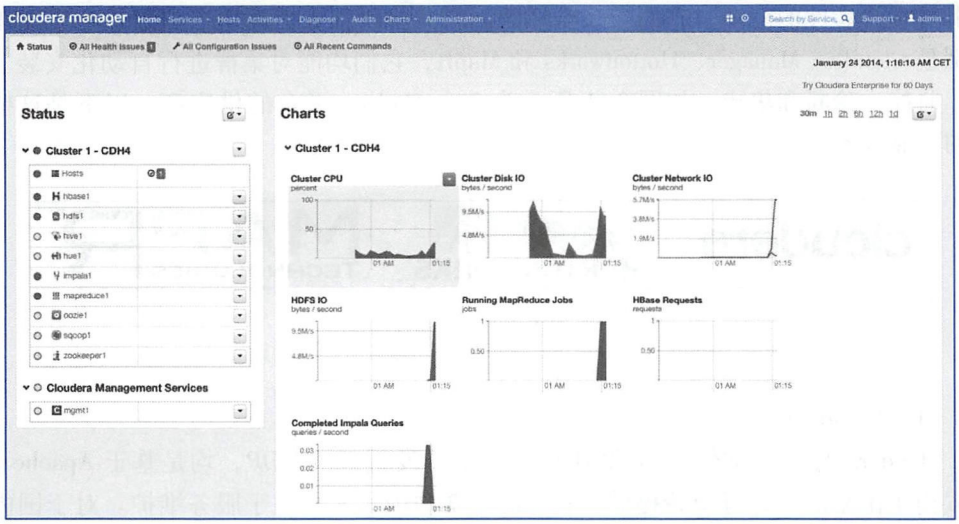



图 2-2 Cloudera Manager 管理界面

（2）Hortonworks

Hortonworks 是唯一一家专心致力于将自主研发 HDP 和 HDF 平台开源到底的公司。Hortonworks 这个名字源自 Horton 系列儿童书中一只叫作 Horton 的大象。众所周知，Hadoop 的名字取自一只毛绒玩具象。类似的取名方式说明 Hortonworks 围绕 Hadoop 展开业务。

2011 年，雅虎公司将 Hadoop 业务剥离出来，由雅虎公司主导 Hadoop 开发的副总裁 Eric Bladeschweiler 带领 20 多个核心成员成立 Hortonworks。成立伊始，Hortonworks 即获得雅虎公司和 Benchmark 公司 2300 万美元的投资。此后 Hortonworks 一直受到资本市场追捧。

Hortonworks 有 HDP 和 HDF 两款核心产品，都是企业级数据管理平台。HDP 基本涵盖了 Hadoop 生态系统的所有组件，包括 HDFS、MapReduce、YARN、Hive、HBase、ZooKeeper、Spark、Flume、Oozie 等。HDF 是 2015 年推出的实时数据流管理平台，是对 HDP 的补充。不同于传统软件提供商，像 Clouder 的 CDH 存在部分产品版本收费，Hortonworks 则没有对产品收费，而是将这两款产品完全开放，将核心技术放在 Hadoop 开源社区中，每个用户都可以看到并使用这两款产品。

那么 Hortonworks 的收益来源是什么？对于企业客户来说，就算知道新技术的核心内容，具体应用还是会碰到很多问题，例如，如何与自己系统相结合、增强功能、调试故障、对接应用等。Hortonworks 公司便从中看到了商机，通过提供支持服务和后期维护，向企业级客户收费。Hortonworks 将核心技术完全公开，用于推动 Hadoop 社区的发展，这样做的好处是，如果开源技术有很大提升，则其将受益最大，因为定制化程度较少，自身不会受到技术提升的冲击。

其中 IBM 公司在 Hadoop 分发版中采用 HDP，并将其与 Data Science Experience 和机器学习全面集成。因此，该解决方案将为客户提供 HDP 丰富的数据安全性、监管

和操作功能，以及 Data Science Experience 的高级分析和管理功能。IBM 把现有的 IBM BigInsights 用户迁移到 HDP。如图 2-3 所示为 HDP 资源栈。

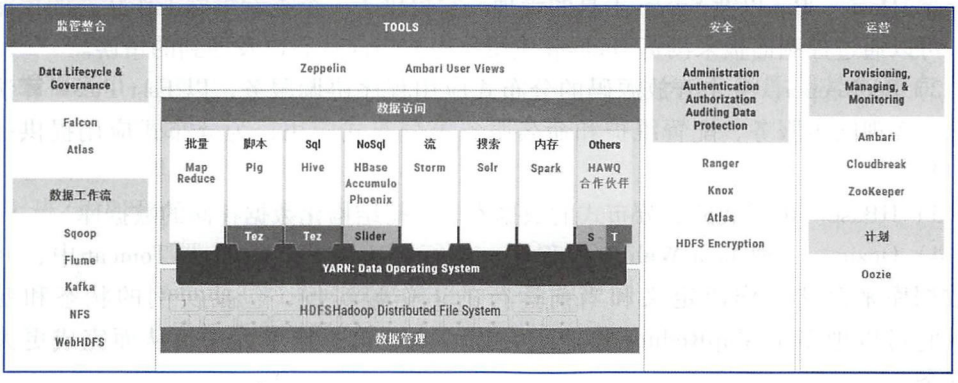



图 2-3　HDP 资源栈

（3）MapR

MapR 是 MapR Technologies 的一个产品，称为下一代 Hadoop，致力于使 Apache Hadoop 成为一个高速、高可靠性、易于管理、使用更加方便的分布式计算服务和存储平台。它极大地扩大了 Hadoop 的使用范围和方式，包含了开源社区许多流行的工具和功能，如 HBase、Hive，还 100%和 Apache Hadoop 的 API 兼容，能够为客户节约一半的硬件资源消耗，使更多的组织和公司能够利用海量数据分析的力量提高竞争优势。

除了这 3 家公司提供的产品外，还有很多公司也提供了相应的数据平台产品，包括 Amazon EC2（Elastic Compute Cloud）、Datameer 的 DAS（Datameer Analytics Solution）产品、EMC Greenplum 统一分析平台（UAP）等。

### 2. 熟悉 Hadoop 生态系统

Hadoop 是一个开源框架，可编写和运行分布式应用处理大规模数据。Hadoop 框架的核心是 HDFS 和 MapReduce。其中 HDFS 是分布式文件系统，MapReduce 是分布式数据处理模型和执行环境。Hadoop 生态系统如图 2-4 所示。

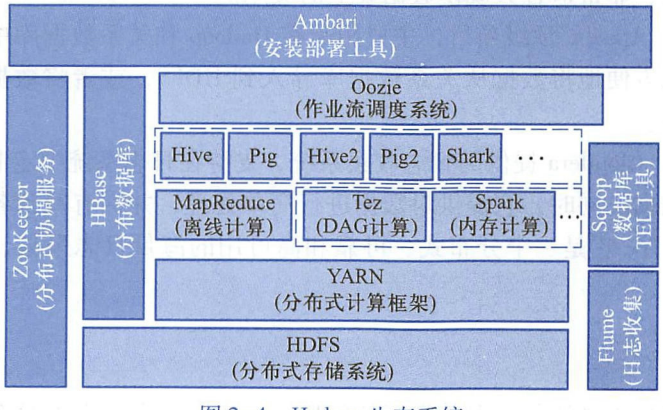



图 2-4　Hadoop 生态系统

笔记

1）Ambari。基于 Web 的 Apache Hadoop 集群管理工具，提供对集群部署、监控和管理。同样也支持对 HDFS、Hadoop MapReduce、Hive、HCatalog、HBase、ZooKeeper、Oozie、Pig 以及 Sqoop 工具的管理。它提供了一个查看集群工作状态的控制面板，可以通过控制面板来浏览 Hadoop 应用程序的当前状态以及遇到的错误。

2）ZooKeeper。一个开放源码的分布式应用程序协调服务，以 Fast Paxos 算法为基础，实现同步服务、配置维护和命名服务等分布式应用，为分布式应用提供一致性 HA。

3）HBase。可扩展的、分布式的数据库，支持结构化数据存储的数据库。

4）Oozie。一种 Java Web 应用程序，运行在 Java Servlet 容器 Tomcat 中，并使用数据库来存储工作流定义和当前运行的工作流实例，包括实例的状态和变量等。它可以把多个 MapReduce 作业组合到一个逻辑工作单元中，从而完成更大型的任务。

5）Hive。提供数据汇总和临时查询的数据仓库基础结构。

6）Pig。一种高级并行计算数据流语言。

7）Shark。即 Hive on Spark，本质上是通过 Hive 的 HQL 解析，把 HQL 翻译成 Spark 上的 RDD 操作，然后通过 Hive 的 metadata（元数据）获取数据库里的表信息，实际 HDFS 上的数据和文件会由 Shark 获取并放到 Spark 上运算。

8）MapReduce。一个软件框架，基于该框架能够容易地编写应用程序，这些应用程序能够运行在由上千个商用机器组成的大集群上，并以一种可靠的、具有容错能力的方式并行地处理 TB 级别的海量数据集。

9）Tez。Apache 最新的支持 DAG 作业的开源计算框架，可以将多个有依赖的作业转换为一个作业，从而大幅提升 DAG 作业的性能。

10）Spark。整个 BDAS 的核心组件，是一个大数据分布式编程框架，不仅实现了 MapReduce 的算子 map 函数和 reduce 函数及计算模型，还提供更为丰富的算子，如 filter、join、groupByKey 等。它是一个用来实现快速而通用的集群计算的平台。

11）YARN。被用于任务调度以及集群资源管理的框架。

12）HDFS。一个高度容错性的系统，适合部署在廉价的机器上。它能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。

13）Sqoop。Apache 顶级项目，主要用来在 Hadoop 和关系数据库中传递数据。通过它，用户可以方便地将数据从关系数据库导入到 HDFS，或者将数据从 HDFS 导出到关系数据库中。

14）Flume。Cloudera 提供的日志收集系统，支持在日志系统中定制各类数据发送方，用于收集数据。同时，它提供对数据进行简单处理，并拥有写到各种数据接受方（可定制）的能力。它是一个分布式、可靠和高可用的海量日志采集、聚合和传输的系统。

微课：2-1 Ambari Hadoop 系统搭建

## 任务实施

虽然大数据越来越流行，但其学习的门槛却一直阻碍着很多的分布式应用初学者

或者大数据的业务应用开发者。多个产品之间会产生不兼容问题，快速集成和维护也显得比较困难。不管是 Hadoop V1 或者 V2 的安装，又或者 Spark、YARN 等的集成，都不是几行简单命令可以完成的，而是需要手工修改很多的集群配置，这进一步增加了业务开发者的学习和使用难度。有了 Ambari，这些问题迎刃而解。

Ambari 管理平台通过安装向导来进行集群的搭建，简化了集群供应。同时，它还有一个监控组件 Ambari-Metrics，可以提前配置好关键的运维指标（metrics），然后收集集群中的服务、主机等运行状态信息，通过 Web 的方式显示出来。用户可以直接查看 Hadoop Core（HDFS 和 MapReduce）及相关项目（如 HBase、Hive 和 HCatalog）是否健康。其用户界面非常直观，用户可以轻松有效地查看信息并控制集群。本任务就是实践 Ambari 平台的构建，实现 Hadoop 分布式集群配置与管理。

Ambari 支持作业与任务执行的可视化与分析，能够更好地查看依赖和性能。通过一个完整的 RESTful API 把监控信息显示出来，集成了现有的运维工具。平台使用 Ganglia 收集度量指标，利用 Nagios 支持系统报警。

Ambari 是一个分布式架构的软件，主要由 Ambari Server 和 Ambari Agent 两部分组成，如图 2-5 所示。Ambari Server 会读取集群中相应服务的配置文件。当用户使用 Ambari 创建集群时，Ambari Server 传送相应的配置文件以及服务生命周期的控制脚本到 Ambari Agent。Agent 得到配置文件后，会下载并安装相应的服务，Ambari Server 会通知 Agent 去启动和管理服务。之后 Ambari Server 会定期发送命令到 Agent 检查服务的状态，将状态信息上报给 Server，并呈现在 Ambari 的 GUI 上，方便用户了解到集群的各种状态，并进行相应的维护。

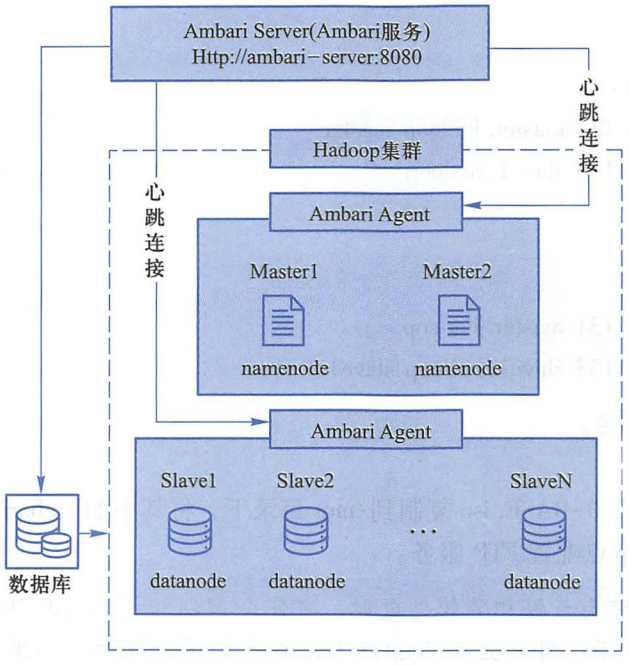



图 2-5　Ambari Server 架构

（1）基本环境配置

本任务以两台节点为例来组建 Hadoop 分布式集群，这里系统版本采用的是 CentOS 7，具体要求见表 2-1。

表 2-1 集群主机规划

| 主 机 名 | 内 存 | 硬 盘 | IP 地址 | 角 色 |
|---|---|---|---|---|
| master | 8 GB | 100 GB | 192. 168. 200. 131 | Ambari-Server、Ambari-Agent、MySQL |
| slave1 | 4 GB | 100 GB | 192. 168. 200. 133 | Ambari-Agent |

根据表中的资源配置来组建大数据基础平台。

① 配置主机名。

```
# master
# hostnamectl set-hostname master
# hostname
master
# slave1
# hostnamectl set-hostname slave1
# hostname
slave1
```

② 修改 hosts 映射文件。

```
# master
# vi /etc/hosts
192. 168. 200. 131 master. hadoop master
192. 168. 200. 133 slave1. hadoop
# slave1
# vi /etc/hosts
192. 168. 200. 131 master. hadoop
192. 168. 200. 133 slave1. hadoop slave1
```

工具 2-1：BigData-v2. 0-BASE 镜像

③ 配置 Yum 源。

```
# master
```

将 BigData-v2. 0-BASE. iso 复制到/mnt 目录下，将其中的 ambari 解压到/opt 目录下，并在 master 节点配置 FTP 服务。

工具 2-2：CentOS 7 镜像

**注意：** 在安装大数据相关软件包时，可能会用到相关依赖软件包，因此需要配置 CentOS 7 Yum 源。可以配置 CentOS 7 的本地 Yum 源文件，如果有网络也可以配置公网的 Yum 源。这里可以采用本地 CentOS 7 Yum 源。

# master& slave1

```
# cd /etc/yum. repos. d/
# rm -vf *
```

**注意**：如果配置公网 Yum 源，则不需要删除原来的 REPO 文件。

编写 REPO 文件，指定 Ambari 安装包路径和 CentOS 7 安装包路径。

```
# vi ambari. repo
[centos7]
baseurl=ftp://192. 168. 2. 10/centos7/
(注:具体的 Yum 源根据真实环境配置,本次为实验室测试环境)
gpgcheck=0
enabled=1
name=centos
[ambari]
name=ambari
baseurl=ftp://10. 0. 3. 61/ambari
(注:具体的 Yum 源根据真实环境配置,本次为实验室测试环境)
enabled=1
gpgcheck=0
```

# master

```
# yum -y install httpd
```

将/mnt/目录下的 HDP-2.4-BASE 和 HDP-UTILS-1.1.0.20 两个文件夹复制到/var/www/html/目录下。启动 httpd 服务。

```
# systemctl enable httpd. service
# systemctl status httpd. service
```

④ 配置 ntp。

# master

```
# yum -y install ntp
# vi /etc/ntp. conf
(注:注释或者删除以下 4 行)
server 0. centos. pool. ntp. org iburst
server 1. centos. pool. ntp. org iburst
server 2. centos. pool. ntp. org iburst
server 3. centos. pool. ntp. org iburst
(注:添加以下两行)
server 127. 127. 1. 0
```

```
fudge 127. 127. 1. 0 stratum 10
#systemctl enable ntpd
#systemctl start ntpd
```

# slave1

```
# yum -y install ntpdate
# ntpdate master. hadoop
# systemctl enable ntpdate
```

⑤ 配置 SSH。

# master& slave1

检查两个节点是否可以通过无密钥相互访问，如果未配置，则进行 SSH 无密码公钥认证配置。

```
# yum install openssh-clients
# ssh-keygen
# ssh-copy-id master. hadoop
# ssh-copy-id slave1. hadoop
```

ssh 登录远程主机查看是否成功。

```
# ssh master. hadoop
# exit
# ssh slave1. hadoop
# exit
```

⑥ 禁用 Transparent Huge Pages。

操作系统后台的 khugepaged 进程会一直扫描所有进程占用的内存，在可能的情况下会把 4k page 交换为 Huge Pages。在该过程中，对于操作的内存的各种分配活动都需要各种内存锁，直接影响程序的内存访问性能，且该过程对于应用是透明的，在应用层面不可控制，对于专门为 4k page 优化的程序来说，可能会造成随机的性能下降现象。

# master& slave1

```
# cat /sys/kernel/mm/transparent_hugepage/enabled
[always] madvise never
# echo never > /sys/kernel/mm/transparent_hugepage/enabled
# echo never > /sys/kernel/mm/transparent_hugepage/defrag
# cat /sys/kernel/mm/transparent_hugepage/enabled
always madvise [never]
```

重启后失效，需要再次执行。

⑦ 安装配置 JDK。

# master

```
# mkdir /usr/jdk64/
# mount -o loop XianDian-BigData-v2.0.5-BASE.iso /mnt/
# cd /mnt/
# tar -zxvf jdk-8u77-linux-x64.tar.gz -C /usr/jdk64/
# vi /etc/profile
export JAVA_HOME=/usr/jdk64/jdk1.8.0_77
export PATH=$JAVA_HOME/bin:$PATH
# source /etc/profile
#Java -version
Java version "1.8.0_77"
Java(TM) SE Runtime Environment (build 1.8.0_77-b03)
Java HotSpot(TM) 64-Bit Server VM (build 25.77-b03, mixed mode)
```

# slave1

```
#mkdir /usr/jdk64/
#scp 172.24.14.106://mnt/jdk-8u77-linux-x64.tar.gz.
# tar -zxvf jdk-8u77-linux-x64.tar.gz -C /usr/jdk64/
# vi /etc/profile
export JAVA_HOME=/usr/jdk64/jdk1.8.0_77
export PATH=$JAVA_HOME/bin:$PATH
# source /etc/profile
#Java -version
Java version "1.8.0_77"
Java(TM) SE Runtime Environment (build 1.8.0_77-b03)
Java HotSpot(TM) 64-Bit Server VM (build 25.77-b03, mixed mode)
```

(2) 配置 Ambari Server

# master

```
# yum -y installambari-server
```

① 配置 MariaDB 数据库。

# master

安装 MariaDB 数据库。

```
# yum install      mariadb mariadb-server mysql-connector-java
```

启动 MariaDB 数据库。

```
# systemctl enable mariadb
# systemctl start mariadb
```

配置 MariaDB 数据库。

```
# mysql_secure_installation
(注:按〈Enter〉键确认后设置数据库 root 密码,这里设置为"bigdata")
Remove anonymous users? [Y/n] y
Disallow root login remotely? [Y/n] n
Remove test database and access to it? [Y/n] y
Reload privilege tables now? [Y/n] y
```

创建 Ambari 数据库。

```
# mysql -uroot -pbigdata
MariaDB [(none)]> create database ambari;
MariaDB [(none)]> grant all privileges on ambari.* to 'ambari'@'localhost'identified by 'bigdata';
MariaDB [(none)]> grant all privileges on ambari.* to 'ambari'@'%'identified by 'bigdata';
MariaDB [(none)]> use ambari;
MariaDB [ambari]> source /var/lib/ambari-server/resources/Ambari-DDL-MySQL-CREATE.sql
MariaDB [ambari]> Bye
```

② 安装配置 Ambari Server。

```
# master
# ambari-server setup
WARNING: SELinux is set to 'permissive'mode and temporarily disabled. OK to continue [y/n] (y)?
Customize user account for ambari-server daemon [y/n] (n)? n
Checking JDK...
[1] Oracle JDK 1.8+Java Cryptography Extension (JCE) Policy Files 8 [2] Oracle JDK 1.7+Java Cryptography Extension (JCE) Policy Files 7 [3] Custom JDK
==========================================================================
Enter choice (1): 3
Path to JAVA_HOME: /usr/jdk64/jdk1.8.0_77
Validating JDK on Ambari Server...done. Completing setup...
Configuring database...
Enter advanced database configuration [y/n] (n)? y
Configuring database...
==========================================================================
    Choose one of the following options:
```

```
[1]-PostgreSQL (Embedded)
[2]-Oracle
[3]-MySQL
[4]-PostgreSQL
[5]-Microsoft SQL Server (Tech Preview) [6]-SQL Anywhere
==========================================================================
== Enter choice (1): 3
Hostname (localhost): Port (3306):
Database name (ambari): Username (ambari):
Enter Database Password (bigdata):
Proceed with configuring remote database connection properties [y/n] (y)?
Ambari Server 'setup' completed successfully.
```

③ 启动 Ambari Server 服务。

```
# ambari-server start
```

打开浏览器，登录 http://192.168.200.131:8080/，在“用户名”文本框中输入“admin”，在“密码”文本框中输入“admin”，如图 2-6 所示。

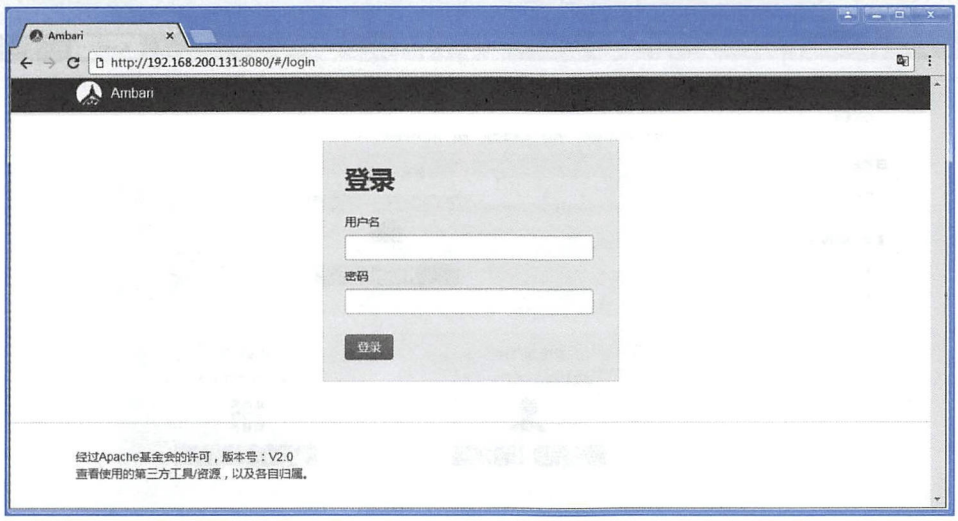



图 2-6　Ambari 登录页面

（3）配置 Ambari Agent

# master& slave1

① 安装 Ambari Agent 并修改 ambari-agent. ini 文件指定 Ambari Server 的地址或者主机名。

```
# yum -y install ambari-agent
# vi /etc/ambari-agent/conf/ambari-agent. ini
```

```
[server]
hostname=master
# ambari-agent restart
```

② 打开 Ambari Agent 的 log 日志文件/var/log/ambari-agent/ambari-agent.log，查看是否存在“Building Heartbeat: {responseId=.....”（如下代码所示）等字样，如果有则表示 Ambari Server 和 Ambari Agent 之间已经成功传递了心跳连接信号。

```
# tail -f /var/log/ambari-agent/ambari-agent.log
INFO 2017-01-12 09:44:20,919 Controller.py:265-Heartbeat response received (id=1340)INFO 2017-01-12 09:44:30,820 Heartbeat.py:78 - Building Heartbeat:{responseId=1340,
timestamp=1484214270820, commandsInProgress=False, componentsMapped=True}
```

（4）部署管理 Hadoop 集群

登录 http://{IPAddress}:8080/，在“用户名”文本框中输入“admin”，在“密码”文本框中输入“admin”。接下来就可以启动安装向导、创建集群和安装服务，如图 2-7 和图 2-8 所示。

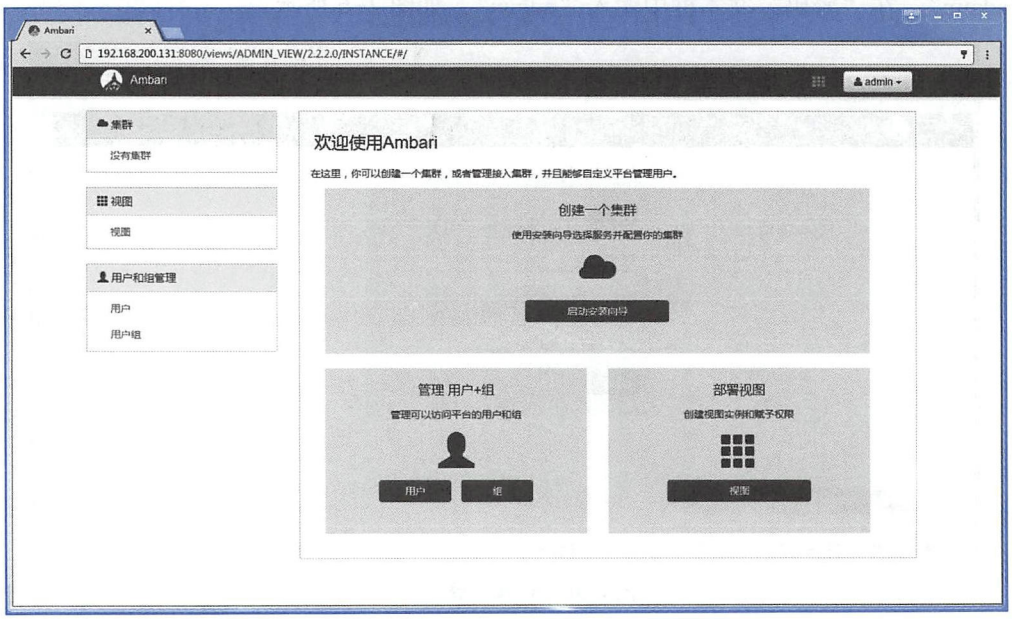



图 2-7 Ambari 管理页面

在“选择安装栈”时指定安装源 HDP-2.4 和 HDP-UTILS-1.1.0.20 的位置。指定相应的目标主机并选择手动注册主机，选择所需要安装的服务，本任务需要安装 HDFS、YARN + MapReduce2、Zookeeper、Ambari Metrics、Hive、HBase、Mahout、Sqoop、Spark 等服务。在整个过程需要设置 Grafana Admin 和 Hive 的密码。部署完成页面如图 2-9 所示。

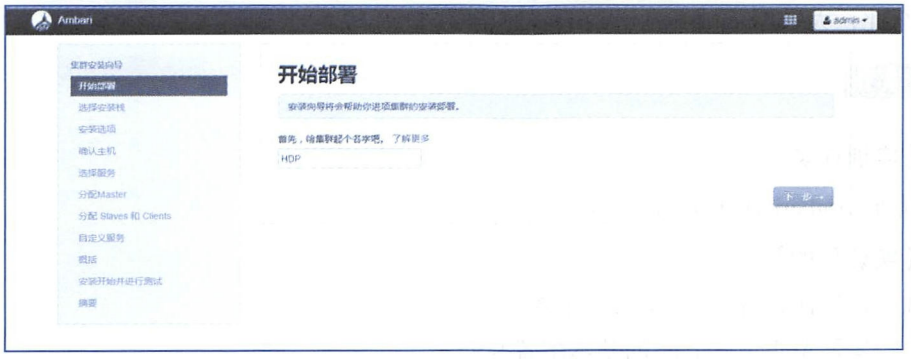



图 2-8　Ambari 安装向导

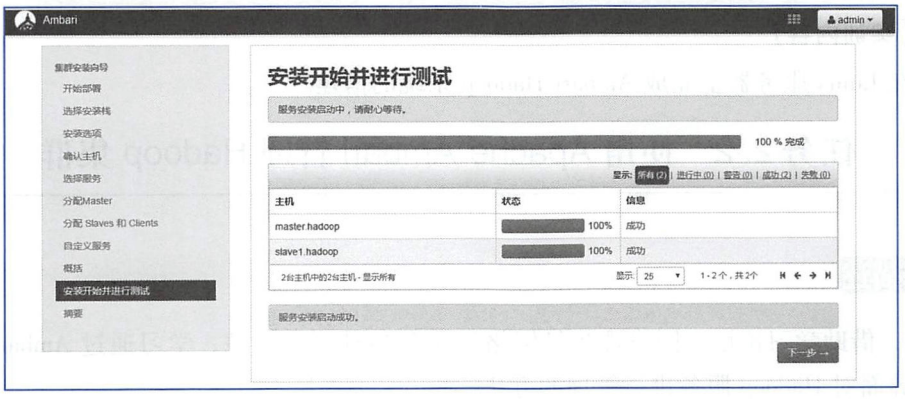



图 2-9　部署完成页面

单击页面导航栏的“主界面”按钮，在主界面中可以查看集群状态和监控信息，如图 2-10 所示。

图 2-10　平台主界面

至此，大数据基础平台已经部署完毕了。

## 项目实训

**【实训任务】**

完成 Ambari Hadoop 系统的搭建。

**【实训目的】**

1. 学会 Hadoop 平台的基本环境配置。
2. 学会 Ambari Server 的基本配置与安装。
3. 学会 Hadoop 集群的部署。

**【实训内容】**

在 Linux 服务器上完成 Ambari Hadoop 系统的搭建。

# 任务 2.2 使用 Apache Ambari 管理 Hadoop 集群

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习通过 Ambari Web 管理界面对 Hadoop 服务进行管理的方法。
2. 学习通过 Ambari Web 管理界面对集群主机进行管理的方法。
3. 学习通过 Ambari Web 管理界面对 Hadoop 服务进程进行管理的方法。
4. 学习通过 Ambari Web 管理界面对服务配置文件进行管理的方法。
5. 使用 Ambari，完成对 Hadoop 集群的管理。

## 任务目标

1. 学会使用 Ambari Web 管理界面对 Hadoop 服务进行管理的方法。
2. 学会使用 Ambari Web 管理界面对集群主机进行管理的方法。
3. 学会使用 Ambari Web 管理界面对 Hadoop 服务进程进行管理的方法。
4. 学会使用 Ambari Web 管理界面对服务配置文件进行管理的方法。

## 知识准备

Ambari 的用户图形界面非常有助于平台管理员去管理维护和监控 Hadoop 集群。

在 Hadoop 集群部署完成后，打开部署 Ambari Server 主机的 8080 端口。默认的管理员用户名为 admin，密码为 admin。登录后进入的是 Ambari 管理的 Hadoop 集群主界面，该界面形象化地展示了集群服务的运行状态、资源使用状况、配置参数以及错误告警等。

## 任务实施

（1）服务管理

在页面左侧的服务列表中，可以选中任何一个想要操作的服务。以 HDFS 为例，在左侧服务列表中选择“HDFS”选项，则在服务列表右侧显示该服务的相关信息，如图 2-11 所示。

图 2-11　HDFS 概要展示页面

选择 HDFS 的“概要”选项卡，显示 HDFS 运行的进程信息，包括运行状态、资源使用情况以及监控信息。单击页面导航栏的“服务”按钮，然后单击导航栏下的“服务操作”按钮，在弹出的下拉菜单中列出很多服务的控制命令，如“启动”“停止”“重启”等。通过这些控制命令，可以对服务进行管理。

Hadoop 的集群部署完成后，并不知道该集群是否可用。此时可以借助“运行服务检查”命令来确保集群服务可以正常运行。选择该命令后，就会在打开的对话框中显示如图 2-12 所示的 HDFS 服务操作进度显示。

该操作其实就是通过往 HDFS 文件系统/tmp 目录中上传一个临时文件来检测系统运行是否正常。当进度条执行完毕后，全绿代表服务运行正常，全红代表服务运行失败，黄色代表出现告警信息。

“服务操作”下拉菜单中的“启动”“停止”“重启所有”3 个命令的含义分别是指启动、停止、重启集群中所有该服务的进程。当执行进度页面打开时，可以查看每个主机进程的操作进度和运行 log，如图 2-13 所示。

微课：2-2 Hadoop 集群管理

笔记

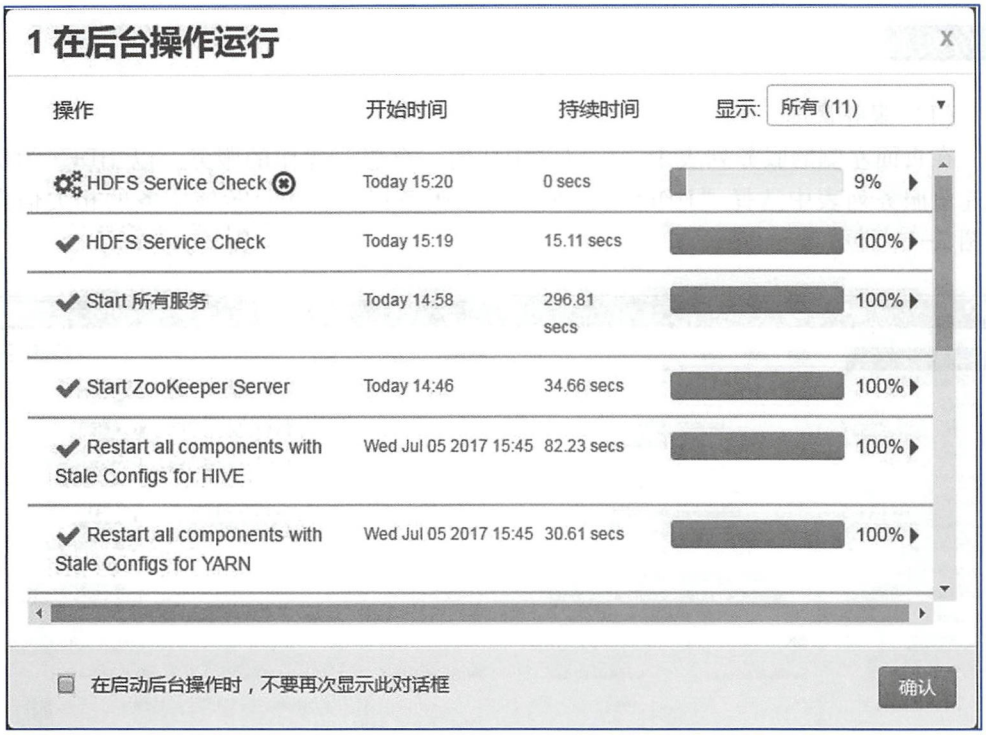



图 2-12 HDFS 服务操作进度显示

图 2-13 HDFS 服务重启界面

“服务操作”下拉菜单中的“打开维护模式”命令是为了让用户在调试或者维护时抑制不必要的告警信息，以及避免批量操作的影响（启动所有服务、停止所有服务、重启所有服务等）。维护模式中有不同的级别设置，分别是服务级别、主机级别以及进程级别。3 种级别之间存在着覆盖关系。例如 HDFS，由于 HDFS 部署在多台主机中，当其维护模式打开后，HDFS 便不会产生任何新的告警。当用户重启集群所有服务时，该服务会越过该批量操作。当用户重启一个机器的所有进程时，该服务的进程也会被越过。

单击“主界面”左侧的服务列表最下方的“动作”按钮，弹出下拉菜单，其中包含有“增加服务”“启动所有服务”和“停止所有服务”等命令。

（2）主机管理

单击页面导航栏中“主机”按钮，打开 Ambari 所管理的主机列表，如图 2-14 所示。

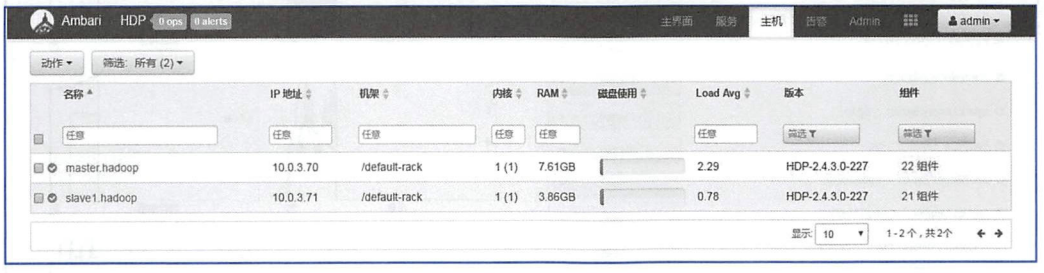



图 2-14　主机列表

单击导航栏左下方的“动作”按钮，在弹出的下拉菜单中列出与主机相关操作的命令，该效果和“服务操作”类似，只是执行的范围不一样，如图 2-15 所示。

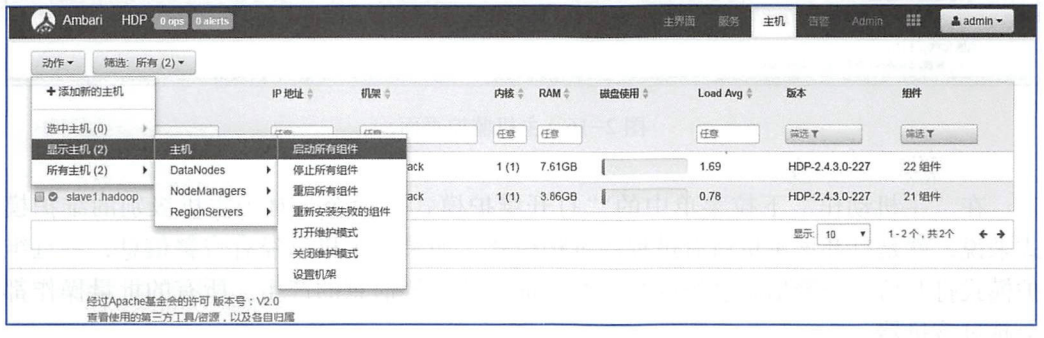



图 2-15　主机动作操作

当用户在“动作”下拉菜单中选择“显示主机”→“主机”→“启动所有组件”命令时，Ambari 就会将所显示主机中的所有服务进行启动。

当用户在“动作”下拉菜单中选择“所有主机”→“DataNodes”→“停止所有组件”命令时，Ambari 就会把所有机器关于 DataNode 的进程关闭。

一旦集群不能够满足生产环境所需的资源时，可以通过“动作”下拉菜单中“添

加新的主机”命令来扩展集群。新的主机节点在加入集群之前，需要完成任务 2.1 中基本环境配置以及 Ambari Agent 服务的安装配置。

进入到其中的一台主机中，如 master，可以看到该主机中所有进程的运行状态、主机资源使用情况、主机的 IP 地址、资源栈等信息。可以在导航栏右下方单击“主机动作”按钮并进行一系列的操作，如图 2-16 所示。

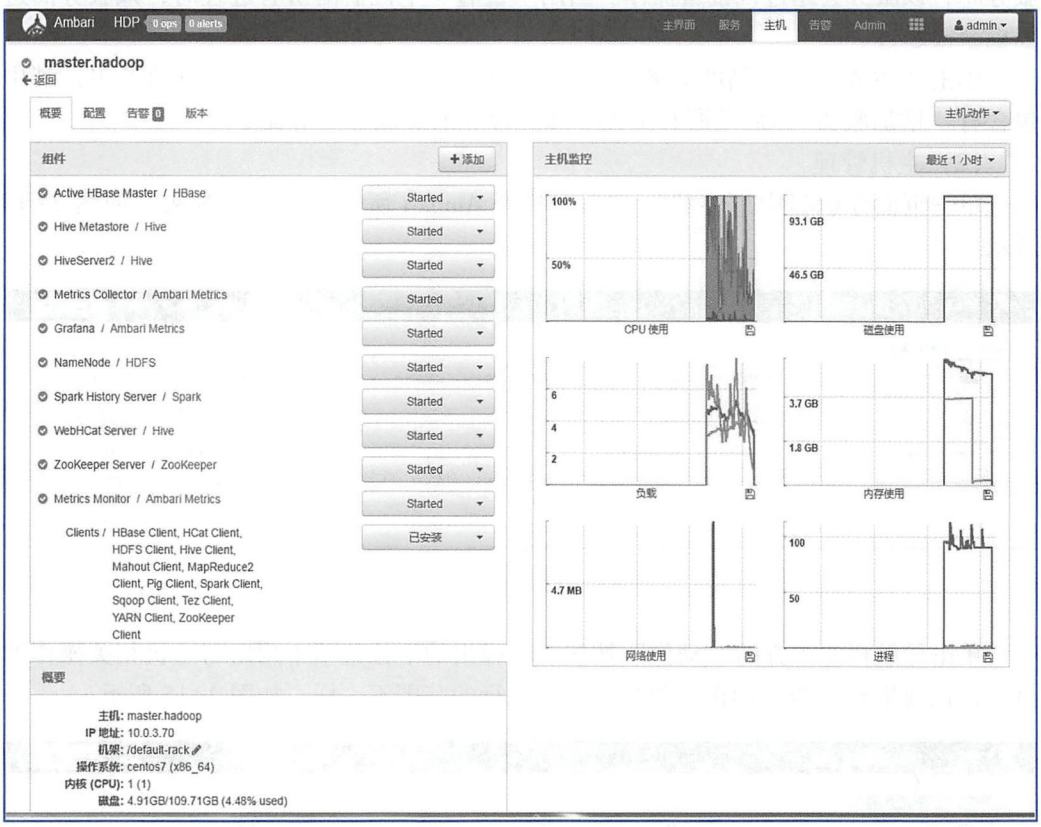



图 2-16 主机监控页面

在“主机动作”下拉菜单中的“打开维护模式”命令，对于主机级别的维护模式来说，就是打开该主机所有进程的维护模式。如果该主机已经有告警信息，一旦维护模式打开后，告警信息会被屏蔽，并且抑制新告警信息的产生。所有的批量操作都会越过该机器。

（3）进程管理

每个服务都由相应的进程组成，如 HDFS 服务，它包含了 NameNode、SNameNode、DateNode 等进程。每台主机中都安装了相应的服务进程，如 master 节点中包含 HDFS 的 NameNode 进程，slave1 节点包含了 SNameNode、DateNode 进程。

进入到 master 节点，找到需要进行管理的进程，如 NameNode，该进程后面有个“Started”按钮，表示该进程正在运行中，单击该按钮可以改变进程的运行状

非卖品，仅供非商业用途或交流学习使用

态，如“重启”“停止”“移动”“打开维护模式”“均衡 HDFS”等，如图 2-17 所示。

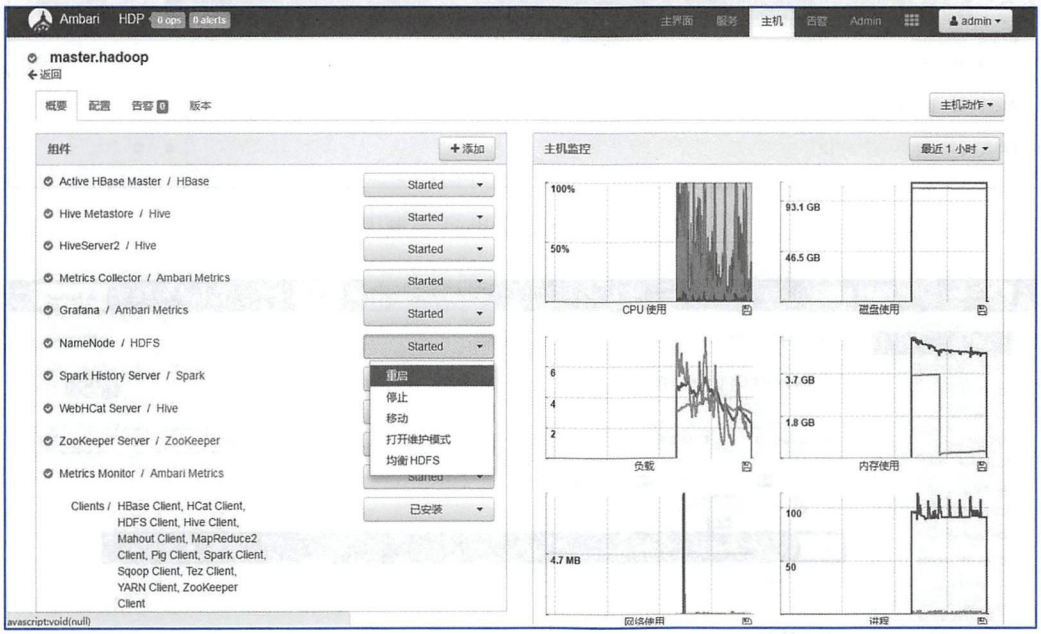



图 2-17　主机进程操作

其中，打开进程级别的维护模式会有两种影响：其一，该进程不再受批量操作的控制；其二，抑制该进程告警信息的产生。例如，打开主机 master 节点的 DataNode 维护模式，那么当用户从“服务操作”下拉菜单中选择“停止”命令，将会停止所有 HDFS 服务，但该主机的 DataNode 不会被关闭，这是因为停止 HDFS 服务的批量操作会直接越过 master 节点中的 DataNode。

笔记

（4）配置管理

Ambari 管理工具可以很方便地修改配置文件并应用到集群的每一台主机中，尤其是当集群中主机的数量非常多时。

例如，需要修改集群 HDFS 文件系统，Block 的复制因子（Block replication）为 2，手动部署集群的情况下，则要修改每一台主机的 hdfs-site. xml 配置文件。如果一个集群有几十台或者几百台主机，这个工作量将非常大。而 Ambari 集群管理工具则可以很好地应对这种情况，由集群中的 Ambari Server 向每台主机中的 Ambari Agent 发送相关的心跳信息，由此更新每台主机中的配置文件。具体操作步骤如下：

在页面导航栏中单击“服务”按钮，在左侧服务列表中选择“HDFS”选项，然后在其服务列表右侧选择“配置”选项卡，选择“Advanced”→“General”→“Block replication”选项，将 Block replication 修改为 2，然后单击“保存”按钮，如图 2-18 所示。

保存成功后，其相应的版本信息如图 2-19 所示，然后单击“重启”按钮，重启

非卖品，仅供非商业用途或交流学习使用

所有标记重启的组件。

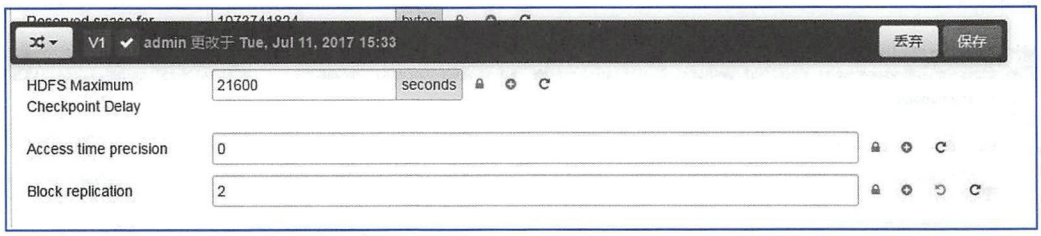



图 2-18 HDFS 修改 Block 复制因子

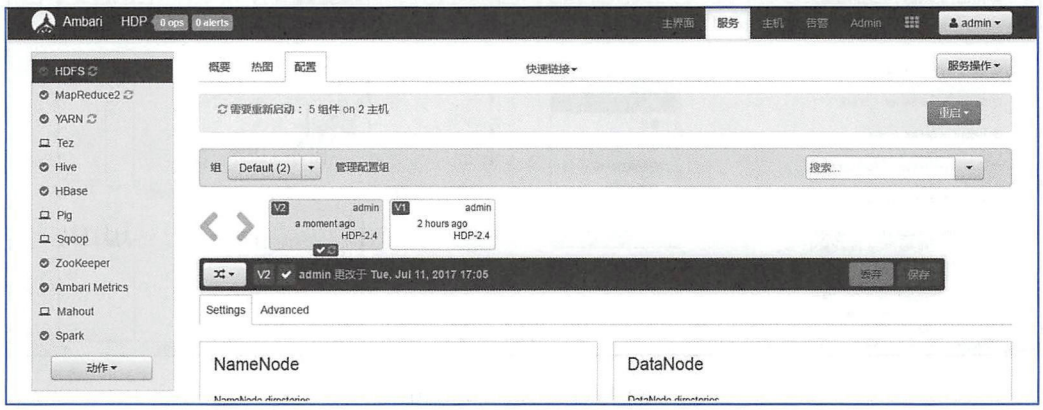



图 2-19 修改后的版本信息

在 Ambari 图形界面中可以查看某一台主机的配置文件。单击页面导航栏的“主机”按钮，在打开的页面中选择相应的主机，选择“配置”标签，便可以查看相应服务的配置信息了。为了保证整个集群配置信息统一，这里禁止单独修改某一台主机的配置文件。

## 项目实训

**【实训任务】**

完成 Ambari Hadoop 系统的 Web 界面管理。

**【实训目的】**

1. 学会使用 Ambari Web 管理界面，对 Hadoop 平台的服务进行管理。
2. 学会使用 Ambari Web 管理界面，对 Hadoop 平台的主机进行管理。
3. 学会使用 Ambari Web 管理界面，对 Hadoop 平台的进程进行管理。
4. 学会使用 Ambari Web 管理界面，对 Hadoop 平台的配置进行管理。

**【实训内容】**

1. 使用 Ambari 的 Web 界面运行 HDFS 服务检查。
2. 使用 Ambari 的 Web 界面在集群中安装 Flume 服务。

非卖品，仅供非商业用途或交流学习使用

3. 使用 Ambari 的 Web 界面尝试打开 WebHCat Server 进程的维护模式。

4. 使用 Ambari 的 Web 界面修改 core-site. xml 配置文件的参数 fs. trash. interval 为 7 小时。

## 项目小结

本项目主要介绍了 Hadoop 基本知识、相关产品及其衍生产品等。了解它们之间的关系和工作模式以及每个产品的特点，学习利用 Ambari 部署工具来构建大数据平台并对集群进行管理的方法，不仅为学情分析系统的实现做好平台准备，而且为今后进行企业大数据平台管理与维护打下坚实的基础。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！

# 项目 3
# 就业岗位数据的采集和存储

PPT3：
就业岗位数据
的采集和存储

## 学习目标

【知识目标】

- 掌握 WebMagic 的流程架构和 HDFS 的基础架构以及 Hive 的基本概念和特点。
- 了解网络爬虫的编程实现步骤和方法。

【技能目标】

- 学会使用 WebMagic 开发网络爬虫的步骤和方法。
- 学会 HDFS 服务进程管理。
- 学会 HDFS 属性配置管理。
- 学会 HDFS 上传文件的方法。
- 学会 Sqoop 上传文件至 Hive 中的方法。

## 学习情境

为了帮助学生制定职业规划和明确就业定向，米老师需要了解当前市场所需人才岗位信息资源，通过学生成绩和岗位技能等信息分析，实现个性化职业技能规划、学习内容和就业岗位推荐。为了获得大量具有时效性的岗位招聘信息，米老师决定编写一个网络爬虫，以便从国内知名招聘网站（智联招聘、51Job、拉勾网等）爬取相关数据信息。在分析相关爬虫开发工具后，选择了 WebMagic 爬虫工具。为了在后续的大数据处理与分析时使用方便，需要将爬取到的相关信息上传到 HDFS 文件系统中，同时需要将学生管理系统中学生成绩的结构化数据上传到 Hive 数据仓库中。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



# 任务 3.1 使用网络爬虫采集岗位招聘数据

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习 WebMagic 爬虫工具的使用方法。

2. 使用 Java 语言编写程序，利用 WebMagic 爬虫工具，从 51Job 网站爬取相关“云计算专业”的岗位招聘信息。

## 任务目标

1. 熟悉 WebMagic 的流程架构。
2. 解析岗位列表页源代码，实现自动添加后续分页链接。
3. 解析岗位信息页源代码，根据条件抓取网页内容。

## 知识准备

笔记

WebMagic 是一个无须配置、便于二次开发的爬虫框架，其提供简单灵活的 API，只需少量代码即可实现一个爬虫。其架构如图 3-1 所示。

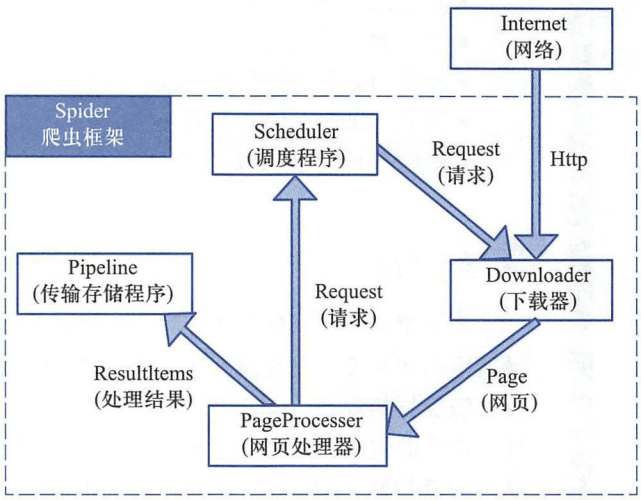



图 3-1 WebMagic 架构图

Downloader 负责从互联网上下载页面，以便后续处理。WebMagic 默认使用 Apache HttpClient 作为下载工具。

PageProcessor 负责解析页面，抽取有用信息，以及发现新的链接。WebMagic 使用 Jsoup 作为 HTML 解析工具，并基于其开发了解析 XPath 的工具 Xsoup。在这 4 个组件中，PageProcessor 对于每个站点每个页面都不一样，是需要使用者定制的部分。

Scheduler 负责管理待抓取的 URL，以及一些去重的工作。WebMagic 默认提供

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



JDK 的内存队列来管理 URL，并用集合来进行去重，并支持使用 Redis 进行分布式管理。除非项目有一些特殊的分布式需求，否则无须自己定制 Scheduler。

Pipeline 负责抽取结果的处理，包括计算、持久化到文件、数据库等。WebMagic 默认提供“输出到控制台”和“保存到文件”两种结果处理方案。Pipeline 定义了结果保存的方式，如果要保存到指定数据库，则需要编写对应的 Pipeline。对于一类需求，一般只需编写一个 Pipeline。

WebMagic 项目中还有一些实验性的功能包，目的只是提供一些与外围工具整合的样例。

（1）webmagic-samples

webmagic-samples 包含早期编写的一些爬虫的案例。这些案例有些使用的仍然是老版本的 API。

最新的、精选过的案例可参阅 webmaigc-core 的 us. codecraft. webmagic. processor. example包和 webmaigc-core 的 us. codecraft. webmagic. example 包。

（2）webmagic-scripts

webmagic-scripts 是 WebMagic 对于爬虫规则脚本化的一些尝试，目标是让开发者脱离 Java 语言进行简单、快速的开发，同时强调脚本的共享。对脚本化感兴趣的读者可以参考 webmagic-scripts 简单文档。

（3）webmagic-selenium

webmagic-selenium 是 WebMagic 与 Selenium 结合的模块。Selenium 是一个模拟浏览器进行页面渲染的工具，而 WebMagic 依赖 Selenium 进行动态页面的抓取。

（4）webmagic-saxon

webmagic-saxon 是 WebMagic 与 Saxon 结合的模块。Saxon 是一个 XPath、XSLT 的解析工具，而 WebMagic 依赖 Saxon 进行 XPath 2.0 语法解析支持。

（5）webmagic-avalon

webmagic-avalon 是一个特殊的项目，它是基于 WebMagic 实现的一个产品化工具，涵盖爬虫的创建、管理等后台工具。Avalon 是亚瑟王传说中的“理想之岛”，webmagic-avalon 的目标是提供一个通用的爬虫产品。

## 任务实施

微课：3-1 使用 WebMagic 采集岗位招聘数据

（1）解析岗位列表页源代码

定制基于 Java 的 WebMagic 爬虫框架抓取 51job 官网上根据关键词“云计算”搜索出的岗位结果。打开 51job 网页搜索“云计算”，显示出来的岗位结果信息页的结构如图 3-2 所示。这是搜索结果的首页，也是爬虫抓取的起始页。可以通过添加后续链接的 URL 至爬取队列，使得爬虫按照要求一步步循序下去。这里需要添加的链接便是具体岗位的信息页，以及后续列表分页。

代码 31：BigDataStudyAnalysis——项目 3

这是在 WebMagic 框架中的 PageProcessor 组件中定制实现的，首先要做的是设置抓取的基本配置，包括编码、抓取间隔、重试次数，代码如下：

```
private Site site = Site.me().setRetryTimes(3).setSleepTime(10).setCharset("gbk");
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！

图 3-2 岗位列表页结构

这里选择重试次数为 3 次，抓取间隔为 10 ms，编码根据为 51job 网页源代码可以查看得到，代码如下：

```
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta http-equiv="Content-Type" content="text/html; charset=gbk">
<link rel="icon" href="/favicon.ico" type="image/x-icon"/>
<title>【云计算招聘,求职】-前程无忧</title>
```

该招聘网站的编码为 gbk。

接下来，判断当前分析的页面是否为岗位列表页。不难发现，列表页的 URL 中都含有 search.51job.com 字符段，可以通过简单的 if 语句判断：

```
if (page.getUrl().toString().contains("search.51job.com"))
```

下面分析岗位信息页链接的 HTML 源代码，并用 Xpath 语法解析出，添加至抓取队列。

检查某个链接源代码的方法是，在岗位列表中右击一项，在弹出的下拉菜单中选择“Inspect”命令，如图 3-3 所示。

其打开该链接的 HTML 源码，如图 3-4 所示。

可以发现，所需要的 URL 地址是一个<p>标签下的唯一的超链接，该标签的 class 名为 t1。同理，该岗位列表页下的所有岗位信息页的 URL 都有着相同的格式，因此可

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



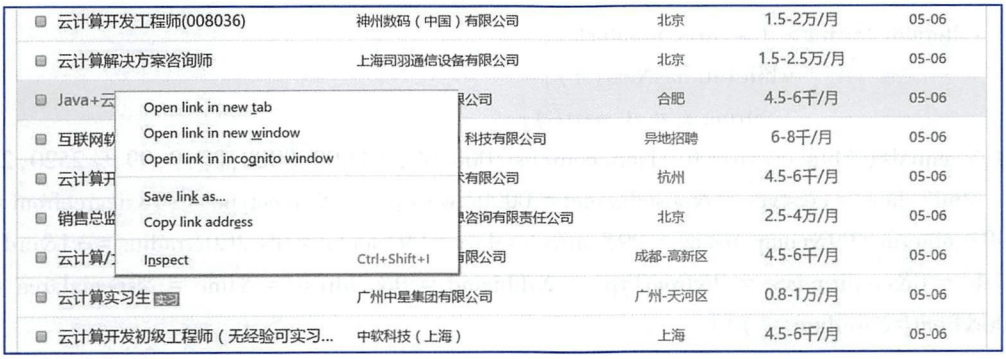



图 3-3　选择“Inspect”命令

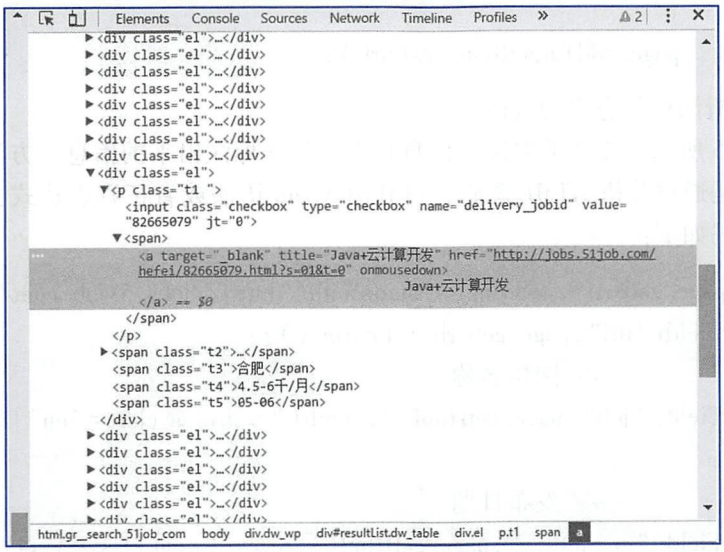



图 3-4　查看 HTML 源代码

以使用 Xpath 全部识别，并添加至抓取队列，实现代码如下：

```
select = page.getHtml().xpath("//p[@class='t1']");
urls = select.links().all();
page.addTargetRequests(urls);
```

添加后续分页的链接也是采用相同的方法，在此省去分析 HTML 源码的截图，直接给出代码：

```
select = page.getHtml().xpath("//div[@class='dw_page']");
urls = select.links().all();
```

需要注意的是，要防止添加空白搜索结果页的链接，否则将会把整个 51job 的岗位全都抓取下来。尽管按照定制的抓取逻辑，这种意外不会出现，但还是添加了踢出队列的保护机制，代码如下：

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
    Iterator<String> it = urls.iterator();
                    while(it.hasNext()){
                        String x = it.next();
if(x.equals("http://search.51job.com/list/000000,000000,0000,00,9,99,%2520,2,
1.html? lang=c&stype=&postchannel=0000&workyear=99&cotype=99&degreefrom=
99&jobterm=99&companysize=99&providesalary=99&lonlat=0%2C0&radius=-1&ord_
field=0&confirmdate=9&fromType=&dibiaoid=0&address=&line=&specialarea=
00&from=&welfare=")){
                            it.remove();
                        }
                    }
                    page.addTargetRequests(urls);
```

（2）解析岗位信息页源代码

如图 3-5 所示，需要抓取岗位信息页上用红框标注出来的信息。方法与上一步中的类似，也是通过分析 HTML 源码，再利用 Xpath 语法或者正则表达式去定位相关内容，具体代码如下：

```
    else if (page.getUrl().toString().startsWith("http://jobs.51job.com/")) {
    page.putField("url",page.getUrl().toString());
                        // 岗位名称
    page.putField("job",page.getHtml().xpath("//div[@class='cn']/h1/text()")
.toString());
                        // 发布日期
    page.putField("createtime",page.getHtml().regex("<em class=\"i4\"></em>([^
<]+)</span>").toString());
                        // 岗位描述
                        page.putField("content",page.getHtml().xpath("//div[@class=
'bmsg job_msg inbox']/html()").toString());
                        // 其他可选项
                         page.putField("salary",page.getHtml().xpath("//div[@class=
'cn']/strong/text()").toString());
                         page.putField("location",page.getHtml().xpath("//span[@
class='lname']/text()").toString());
                         page.putField("company",page.getHtml().xpath("//p[@class=
'cname']/a/text()").toString());
                        page.putField("experience",page.getHtml().regex("<em class=
\"i1\"></em>([^<]+)</span>").toString());
```

版权所有，严禁传播，违者自负法律责任！



```
                page.putField("education",page.getHtml().regex("<em class=\"
i2\"></em>([^<]+)</span>").toString());
                page.putField("number",page.getHtml().regex("<em class=\"i3
\"></em>([^<]+)</span>").toString());}
```

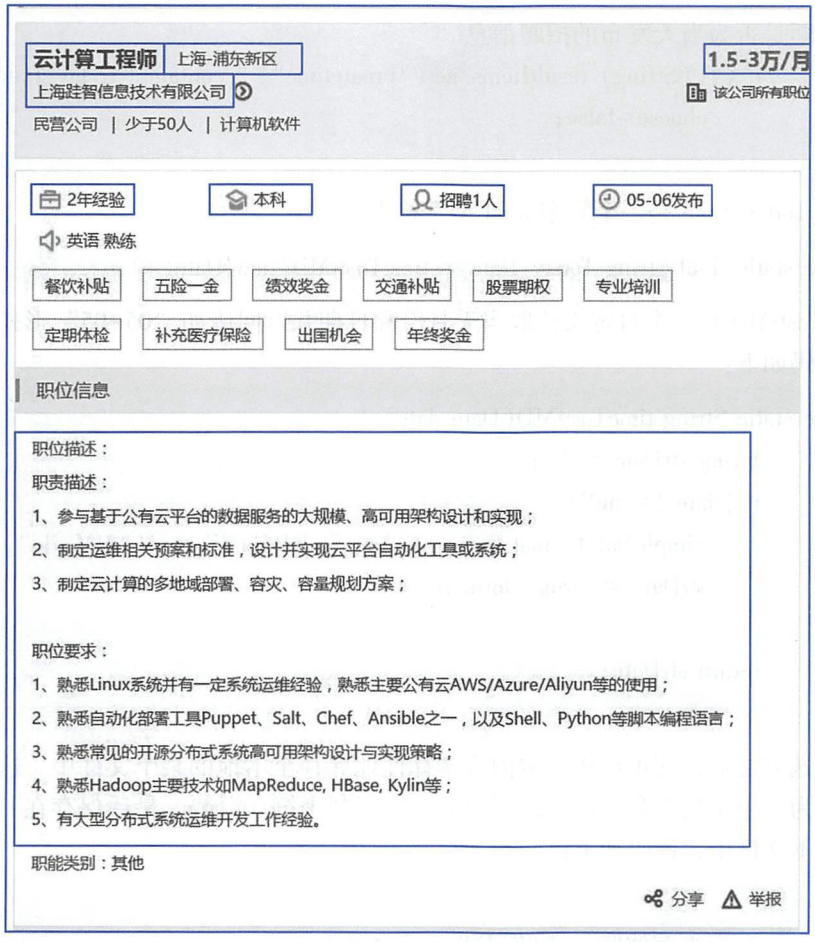



图3-5　岗位信息页结构

（3）根据条件保存抓取到的信息

这里根据条件筛选岗位信息，比如已经列出的一些培训机构的“公司黑名单”，保存在了Dictionary类中的Blacklist静态String数组中，那么可以使用简单的if语句来判断，其代码如下：

```
Boolean choose = true;
//判断是否为培训公司发布的“垃圾”招聘信息
        for (String i:Dictionary.Blacklist){
            if ( ((String) resultItems.get("company")).contains(i)){
```

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



```
                choose=false;
            }
        }
```

对于发布日期，仅需要保留当天发布的招聘信息，判断代码如下：

```
//判断是否为当天发布的招聘信息
        if (!((String) resultItems.get("createtime")).contains(Today_Date)){
            choose=false;
        }
```

其中 Today_Date 是一个静态 final 型字符串：

```
public static final String Today_Date = timeTostrMD(new Date());
```

timeTostrMD 是一个自定义获取当天月份和日期并返回诸如“05-05”形式字符串的方法。代码如下：

```
public static String timeTostrMD(Date date) {
        String strDate = "";
        if (date != null) {
            SimpleDateFormat format = new SimpleDateFormat("MM-dd");
            strDate = format.format(date);
        }
        return strDate;
}
```

在筛选完之后，便可以将抓取内容选择性地保存至本地的某个文件中。这里定制的保存格式为一个岗位保存一行，每行中的信息用制表符\t 分隔，最后保存在一个以日期命名的文本文件中，代码如下：

```
try {
        //定义存储路径,以每天为一个文件存储
        String path = this.path + PATH_SEPERATOR + timeTostrYMD(new Date())
+ ".txt";
        File file = getFile(path);
        String str =resultItems.get("url")
                +"\t"
                +resultItems.get("job")
                +"\t"
                + resultItems.get("location")
                + "\t"
                + resultItems.get("company")
```

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



```
                + "\t"
                + resultItems.get("salary")
                + "\t"
                + resultItems.get("experience")
                + "\t"
                + resultItems.get("education")
                + "\t"
                + resultItems.get("number")
                + "\t"
            + resultItems.get("createtime")
                + "\t"
                + replaceBlank(delHTMLTag(resultItems.get("content")
                    .toString()))
                + "\r\n";

        FileUtils.writeStringToFile(file,str,"utf-8",true);
    } catch (IOException e) {
        logger.warn("write file error",e);
    }
```

（4）运行代码程序

WebMagic 的核心组件为 PageProcessor 与 Pipeline。读者可以自己定制这两个组件，而调用这两个核心组件是通过 Spider 类，其简单代码如下：

```
Spider.create(new CrawlJob())
  .addUrl(URL_START)
  .addPipeline(new PipelineJob()).thread(5).run();
```

由于要将 Java 程序导出为 jar 文件，并且需要实现无参传递（使用默认保存路径）和有参传递（使用指定保存路径）两种形式，因此使用 main 方法的代码如下：

```
public static void main(String[] args) throws Exception {
    String URL_START = "http://search.51job.com/list/000000%252C00,000000,
0000,00,9,99,%25E4%25BA%2591%25E8%25AE%25A1%25E7%25AE%2597,2,1.html?
lang=c&degreefrom=99&stype=1&workyear=99&cotype=99&jobterm=99&companysize=
99&radius=-1&address=&lonlat=&postchannel=&list_type=&ord_field=&curr_page=
&dibiaoid=0&landmark=&welfare=";
if (args.length==0){
Spider.create(new CrawlJob())
.addUrl(URL_START)
```

严禁网络传播本PDF，违者责任自负！

```
    .addPipeline(new PipelineJob()).thread(5).run();
}else{
        Spider.create(new CrawlJob())
        .addUrl(URL_START)
        .addPipeline(new PipelineJob(args[0])).thread(5).run();
            }
}
```

## 项目实训

**【实训任务】**

编程实现从智联招聘网站爬取“大数据”相关岗位的招聘信息。

**【实训目的】**

1. 熟悉 WebMagic 爬虫框架。
2. 掌握在 Processor 组件定制爬虫逻辑和在 Pipeline 组件定制保存规则的方法。
3. 掌握使用 Spider 组件调用 Processor、Pipeline 两个组件的方法。
4. 掌握解析 HTML 源代码的方法。

**【实训内容】**

利用 WebMagic 爬取工具，编程实现从智联招聘网站爬取“大数据”相关岗位的招聘信息。

# 任务 3.2 将采集的数据上传到 HDFS

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习 HDFS 文件系统的使用方法。
2. 将利用 WebMagic 爬取工具爬取的招聘岗位信息上传到 HDFS 文件系统中。

## 任务目标

1. 熟悉 HDFS 工作原理。
2. 学会文件系统的属性配置方法。
3. 学会通过多种方式上传文件至 HDFS 文件系统的方法。

## 知识准备

Hadoop 可以将无数台计算机组成集群，向海量数据提供存储和计算。Hadoop 中有一个综合性的文件系统抽象，它提供了文件系统实现的各类接口。HDFS 便是该抽象文

严禁网络传播本PDF，违者责任自负！

笔记

件系统中的一个实例，它是一个统一向用户提供存储服务的分布式文件系统。

HDFS 是基于流数据模式访问和处理超大文件的需求而开发的，是一个高度容错性的系统，适合运行在通用硬件上，也可以部署在廉价的机器上。它能够提供高吞吐量的数据访问，非常适合大规模数据集上的应用。为实现流式读取文件系统数据的目的，HDFS 放宽了一部分 POSIX 约束。

HDFS 体系中采用 Master/Slave 主从架构，如图 3-6 所示。其中，Master 节点运行 NameNode 进程；Slave 节点运行 DataNode 进程。NameNode 在整个文件系统中充当管理者，管理文件系统的命名空间（Namespace），维护整个文件系统的文件目录树及这些文件的索引目录。这些信息以两种形式存储在本地文件系统中，一种是命名空间镜像（Namespace image）；一种是编辑日志（Edit log）。从 NameNode 中可以获得每个文件的 Block 所存储的位置。需要注意的是，这些信息不是永久保存的，NameNode 会在每次启动系统时动态地重建这些信息。当运行任务时，客户端通过 NameNode 获取元数据信息，和 DataNode 进行交互以访问整个文件系统。系统会提供一个类似 POSIX 的文件接口，使用户在编程时无须考虑 NameNode 和 DataNode 的具体功能。

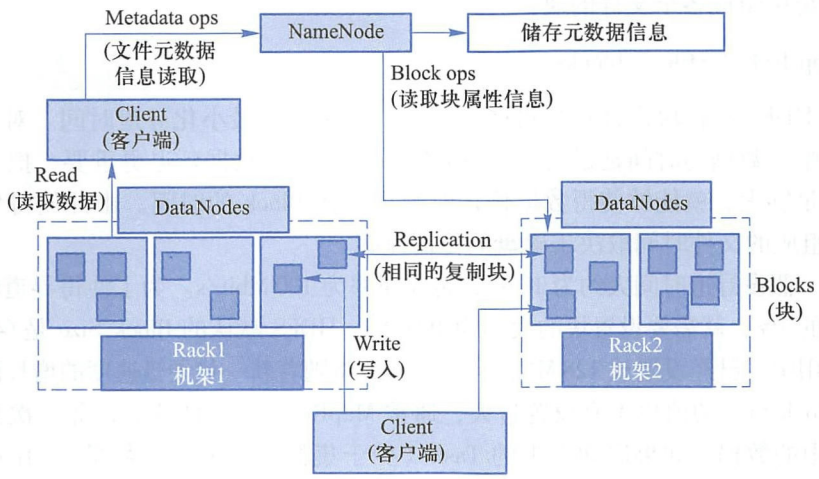



图 3-6　HDFS 体系结构

DataNode 是文件系统的工作节点，用来执行具体的任务，它们根据客户端或 NameNode 的调度存储和检索数据，定期向 NameNode 发送它们所存储的块（Block）的列表。

磁盘中的 Block Size（块）是磁盘读/写数据的最小单位。构建在这样的磁盘上的文件系统也是通过块来管理数据的。文件系统的块通常是磁盘块的整数倍。文件系统的块一般为几千个字节，磁盘块一般为 512 个字节。

HDFS 也有 Block 的概念，但它的块是一个很大的单元，是文件存储处理的逻辑单元，默认是 64 MB。像硬盘中的文件系统一样，在 HDFS 中的文件将会按块大小进行分解，并作为独立的单元进行存储。但和硬盘中的文件系统不同的是，存储在块中的一个比块小的文件并不会占据一个块大小的物理空间（HDFS 中一个块只存储一个文件的内容）。

严禁网络传播本PDF，违者责任自负！

HDFS 作为一个分布式文件系统，是设计用来处理大文件的，使用抽象的块会带来很多好处。一个好处是可以存储任意大的文件，而又不会受到网络中任一单个节点磁盘大小的限制。一般情况下，单个节点存储 100 TB 的数据是不可能的，但由于逻辑块的设计，HDFS 可以将该超大的文件分成众多块，分别存储在集群的各台机器上。另外一个好处是使用抽象块作为操作的单元可简化存储子系统，而简化是所有系统的追求，尤其对频繁出现故障和种类繁多的分布式系统来说，简化就显得尤为重要。在 HDFS 中块的大小固定，这样它就简化了存储系统的管理，特别是元数据信息可以和文件块内容分开存储。不仅如此，块更有利于分布式文件系统中复制容错的实现。在 HDFS 中为了处理节点故障，默认将文件块的副本数设定为 3 份，分别存储在集群的不同节点上。当一个块损坏时，系统会通过 NameNode 获取元数据信息，在另外的机器上读取一个副本并进行存储，这个过程对用户是透明的。当然，这里的文件块副本冗余可以通过文件进行配置，例如在某些应用中，可能会为操作频率较高的文件块设置较高的副本数量以提高集群的吞吐量。

在 HDFS 中，可以通过终端命令直接获得文件和块信息，例如，以下命令可以列出文件系统中组成各个文件的块：

```
hadoop fsck / -files -blocks
```

设计 HDFS 时将块设计得尽可能大一些，这是为了最小化寻道时间。对 HDFS 来说，读取整个数据的时间延迟要比读取到第一条记录的数据延迟更重要，把一个数据块设计得足够大，就能够使得数据传输时减少寻找 Block 的时间。由此使得传输由多个 Block 组成的文件时间取决于磁盘的传输速率。

例如，假设寻道时间大约为 10 ms，传输速率为 100 Mbit/s。为了使得寻道时间仅为传输时间的 1%，就需要设置块的大小为 100 MB。HDFS 默认的 Block Size 是 64 MB，但在实际应用中，已经设置成 128 MB，且这个参数将随着新一代硬盘速度的增长而增长。

但 Block Size 的值也不宜设置过大，通常 MapReduce 中的 Map 任务一次只处理一个 Block 中的数据，如果启动太少的 Task（少于集群中的节点的数量），作业的速度就会比较慢。

微课：3-2 数据上传 HDFS 操作

## 任务实施

（1）服务进程管理

可以通过下列命令查看当前系统运行的进程，如图 3-7 所示。

```
# jps
```

重启 NameNode 和 DateNode 进程。

hadoop-daemon. sh 脚本是启动和停止 Hadoop 后台程序，--config 参数指定启动程序时用到的配置文件目录。大括号中的 namenode 和 datenode 是需要启动的进程，两者择其一。命令如下：

```
1445 AmbariServer
13606 RunJar
22313 HMaster
27226 Jps
21483 HistoryServer
22334 ApplicationHistoryServer
13871 DataNode
28431 NameNode
```

图 3-7 查看系统进程

严禁网络传播本PDF，违者责任自负！

```
# /usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /usr/hdp/current/hadoop-client/conf stop {namenode/datenode}
# /usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /usr/hdp/current/hadoop-client/conf start {namenode/datenode}
```

（2）HDFS 属性配置

可以通过命令行与 HDFS 交互。HDFS 有很多其他接口，但命令行是最简单的，同时也是许多开发者最熟悉的。下面介绍 Hadoop 的属性配置。

1）core-site.xml。设置 Hadoop 默认文件系统。文件系统是由 URI（Uniform Resource Identifier，统一资源标识符）指定的，这里已使用了一个 HDFS URI 来配置 HDFS 为 Hadoop 的默认文件系统。HDFS 的守护程序将通过该属性来决定 HDFS 名称节点的宿主机和端口。它将在 master 节点上运行，端口为 8020。这样，HDFS 用户将通过该属性得知名称节点在哪里运行以便连接到它。

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:8020</value>
</property>
```

设置缓存的大小，不论是对硬盘或者是网络操作来讲，较大的缓存都可以提供更高的数据传输速率，但这也意味着更大的内存消耗和延迟。该参数要设置为系统页面大小的倍数，以 B（字节）为单位。

```
<property>
    <name>io.file.buffer.size</name>
    <value>131072</value>
</property>
```

设置 HDFS 的 NameNode 的格式化信息存储路径。

```
<property>
    <name>hadoop.tmp.dir</name>
    <value>file:/home/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
</property>
```

2）hdfs-site.xml。设置本地文件系统 DFS NameNode 存放 Name Ttable 的路径。

```
<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///usr/hadoop/dfs/name</value>
</property>
```

设置本地文件系统 DFS DataNode 存放数据 Block 的目录。

严禁网络传播本PDF，违者责任自负！

```
<property>
    <name>dfs. datanode. data. dir</name>
    <value>file:///usr/hadoop/dfs/data</value>
</property>
```

设置 HDFS 文件系统 Block 的复制份数。设置份数为 3 时，HDFS 就会按照设置将文件系统块复制为 3 份。如果采用的是伪分布配置，这里参数应设置为 1，否则在单独一个数据节点上运行时，HDFS 无法将块复制到 3 个数据节点上，会持续警告块的副本不够。

```
<property>
    <name>dfs. replication</name>
    <value>3</value>
</property>
```

3）安全模式。NameNode 被启动后，文件系统会进入一个称为安全模式的特殊状态。处于安全模式下的文件系统只可读不可写。NameNode 从所有的 DataNode 上接收心跳信号和块状态报告。块状态报告包括了某个 DataNode 的所有数据块列表。每个数据块都有一个指定的最小副本数。当 NameNode 检测确认某个数据块的副本数目达到最小值时，就会被认为该数据块的副本是安全的，在一定百分比（该参数可配置）的数据块被 NameNode 检测确认安全之后，默认在 30 秒（可通过 dfs. safemode. extension 更改，单位为毫秒）后退出安全模式。接下来它会确定还有哪些数据块的副本没有达到指定数目，并将这些数据块复制到其他 DataNode 上。当然，开发者也可通过命令明确指定进入或退出安全模式。另一种让 NameNode 进入安全模式的方法是，设置 dfs. safemode. threshold. pct 的值大于 1，这样 NameNode 被启动后就会一直停留在安全模式。安全模式在集群升级或维护时非常实用。

当在进行系统维护或者集群维护时，不希望用户再去操作 HDFS 文件系统中的文件，这时候需要手动将 NameNode 设置成安全模式的状态。该操作需要 HDFS 管理用户（即 hdfs 用户）来实现。

进入安全模式的命令如下：

```
$hadoop dfsadmin -safemode enter
$hadoop dfsadmin -safemode get
```

执行结果如下：

```
Safe mode is ON
```

4）文件安全。NameNode 的重要性是显而易见的，没有它客户端将无法获得文件块的位置。在实际应用中，如果集群的 NameNode 出现故障，就意味着整个文件系统中全部的文件会丢失，因为无法再通过 DataNode 上的文件块来重构文件。下面介绍 Hadoop 确保 NameNode 的安全所采用的多种机制。

严禁网络传播本PDF，违者责任自负！

笔记

① 备份 NameNode 上持久化存储的元数据文件，然后将其转储到其他文件系统中。这种转储是同步的、原子的操作。通常的实现方法是将 NameNode 中的元数据转储到远程的 NFS 文件系统中。

② 系统中同步运行一个 Secondary NameNode（二级 NameNode）。该节点的主要作用就是周期性地合并编辑日志中的命名空间镜像，以避免编辑日志过大。Secondary NameNode 的运行通常需要大量的 CPU 和内存去做合并操作，这就要求其运行在一台单独的机器上。在该机器上会存储合并过的命名空间镜像，这些镜像文件会在 NameNode 宕机后作替补使用，以便最大限度地减少文件的损失。需要注意的是，Secondary NameNode 的同步备份总会滞后于 NameNode，所以损失是必然的。

5）可靠性管理。Hadoop 集群中的主机在某些情况下会出现宕机或者系统损坏的问题，一旦遇到这些问题，HDFS 文件系统中的数据文件难免会产生损坏或者丢失，为了保证 HDFS 文件系统的可靠性，可以采取以下策略：

① 冗余副本策略。更改集群冗余副本的复制因子为 5，从而避免某台节点主机损坏导致数据丢失的情况。这里有两种方式来设置复制因子。

第一种方式是通过 Web 界面，进入到 HDFS 配置中修改 Block replication 的参数值，如图 3-8 所示。

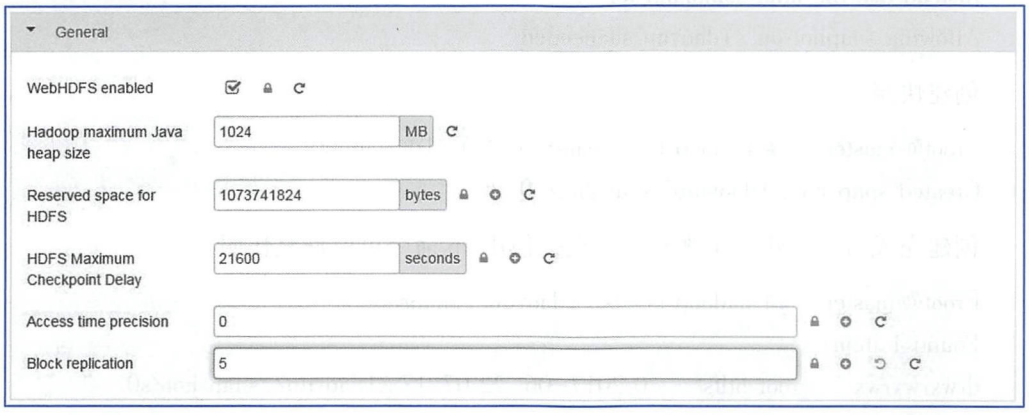



图 3-8　修改 Block replication 的参数值

第二种方式是在 Shell 命令行模式下，修改 hdfs-site.xml 配置文件，将 dfs.replication 的值设置为 5，然后重启 NameNode 和 DataNode 进程。

```
# vi /etc/hadoop/2.3.6.0-3796/0/hdfs-site.xml
        <property>
          <name>dfs.replication</name>
          <value>5</value>
      </property>
```

② 回收站策略。设置 HDFS 文件系统回收站中的文件彻底删除的时间间隔为7 天。进入到 HDFS 配置中修改 fs.trash.interval 的值为 10080（分钟），如图 3-9 所示。

严禁网络传播本PDF，违者责任自负！

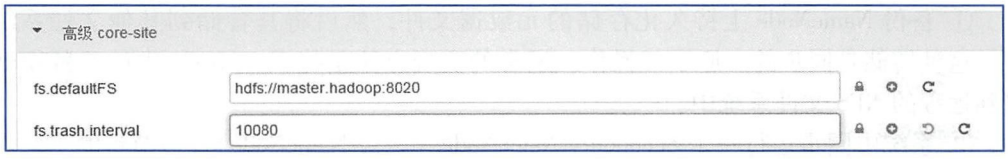



图 3-9 修改 fs. trash. interval 的值

**注意：**当该值为 0 时，表示禁用回收站的功能。

③ 快照管理策略。为 HDFS 文件系统中/1daoyun 目录创建快照，首先需要设置该目录可进行快照功能，然后再创建快照。

```
[root@ master ~]# hadoop fs -ls /1daoyun
Found 5 items
-rw-r--r--   3 root hdfs 46   2017-06-22 07:15 /1daoyun/bigdata. txt
```

首先允许/1daoyun 目录以及子目录可创建快照。

```
[root@ master ~]# hadoop dfsadmin -allowSnapshot /1daoyun
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Allowing snaphot on /1daoyun succeeded
```

创建快照。

```
[root@ master ~]# hadoop fs -createSnapshot /1daoyun s0
Created snapshot /1daoyun/. snapshot/s0
```

创建完成后，创建一个恢复文件夹，使用 cp 命令可以恢复快照。

```
[root@ master ~]# hadoop fs -ls /1daoyun/. snapshot
Found 1 items
drwxrwxrwx   - root hdfs      0 2017-06-22 07:17 /1daoyun/. snapshot/s0
[root@ master ~]# hadoop fs -mkdir /1daoyun/recover
[root@ master ~]# hadoop fs -cp /1daoyun/. snapshot/s0/bigdata. txt /1daoyun/recover
[root@ master ~]# hadoop fs -ls /1daoyun/recover
Found 1 items
-rw-r--r-- root hdfs 46 2017-06-22 07:22 /1daoyun/recover/bigdata. txt
```

（3）Shell 上传采集数据至 HDFS

利用命令行上传采集的数据到/user/PC/dict_in 目录。

工具 3-1：Hadoop 安装包



```
$hadoop fs -put /data/Webmagic/ * /user/PC/dict_in
```

（4）配置案例开发环境

① 运行本案例前，需要对大数据开发环境进行配置。在用于开发的计算机上安装 Hadoop 2. 7. 0，为了方便，解压 hadoop-2. 7. 0. zip 资源包。另外还需要将 hadoop-

2.7.0/bin 目录下的 hadoop.dll 文件复制到系统路径 C:\Windows\System32 下。替换复制完成后，将 Hadoop 安装目录添加到系统环境变量中，首先在用户变量中添加 HADOOP_HOME 变量，如图 3-10 所示。

② 随后在系统变量的 Path 中添加 Hadoop 的 bin 目录路径，如图 3-11 所示。

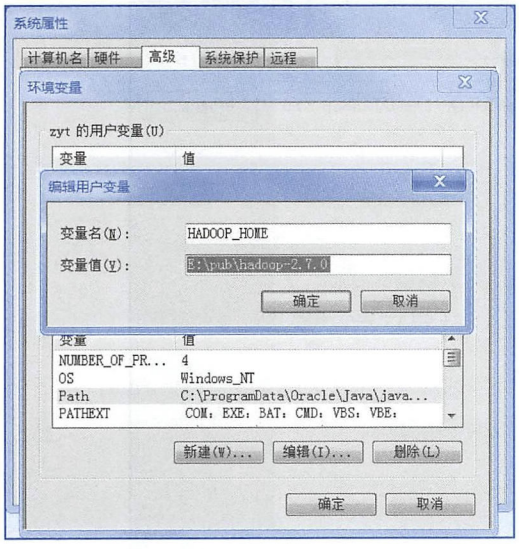



图 3-10　编辑 HADOOP_HOME 变量

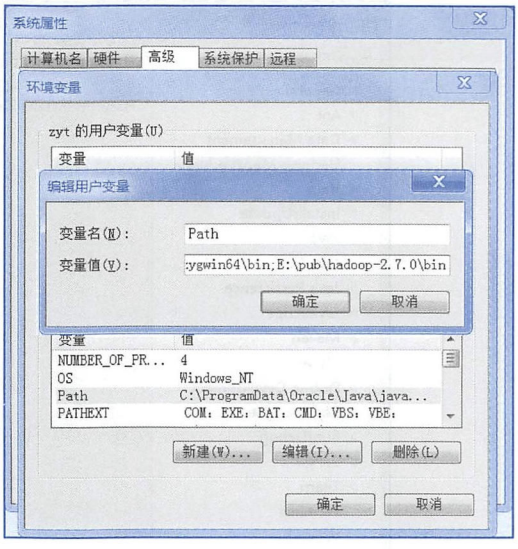



图 3-11　编辑 Path 变量

③ 添加完成后，需要配置 Eclipse，将 Eclipse 的 Hadoop 插件 hadoop-eclipse-kepler-plugin-2.2.0.jar 文件复制到 Eclipse 的 Plugin 目录下，如图 3-12 所示。

工具 3-2：Eclipse 插件

图 3-12　Eclipse 的 Plugin 目录

④ 插件复制完成后，启动 Eclipse，配置 Hadoop 插件。在主菜单栏中选择“Window”→“Preferences”→“Hadoop Map/Reduce”命令，在“Hadoop installation diretory”文本框中输入 hadoop-2.7.0 的目录路径，最后单击“OK”按钮，如图 3-13 所示。

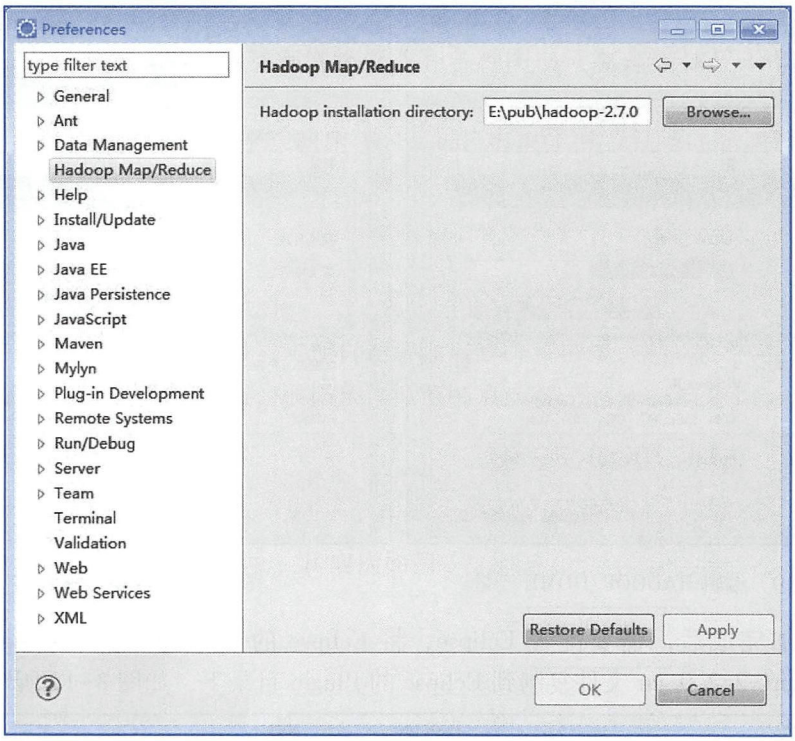



图 3-13 配置 Hadoop 安装路径

⑤ 在 Eclipse 中能看到 DFS Locations，如图 3-14 所示。

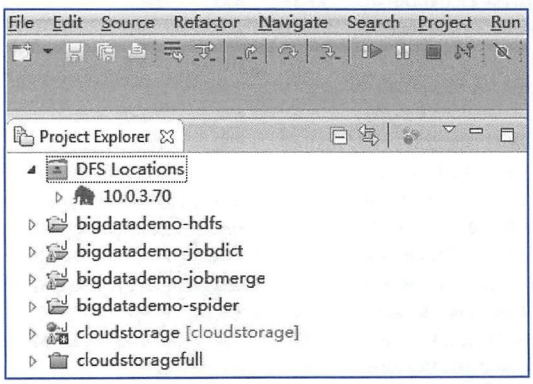



图 3-14 DFS Locations 目录

在 DFS Master 下对应的 Host 以及 Port 文本框中，编辑 HDFS 的访问地址以及端口。这样，Eclipse 就能看到 HDFS 所对应的文件，如图 3-15 所示。

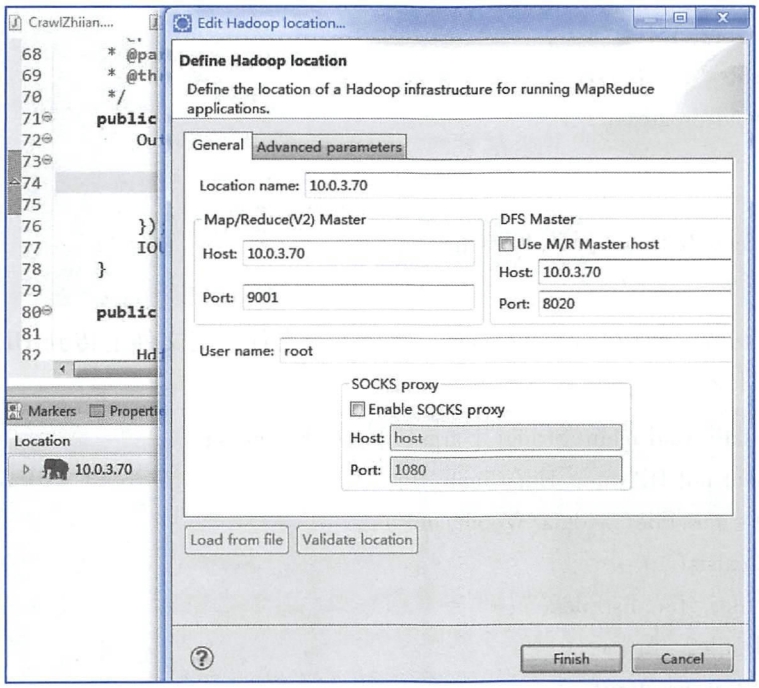



图 3-15　编辑 HDFS 的访问地址与端口

⑥ 编写程序上传数据。导入项目工程 bigdatademo-hdfs，在 HdfsClient 类 中定义 fs. defaultFS。

定义 HDFS 的配置信息：

```
conf=new Configuration();onf=new Conf
conf.set("fs.defaultFS","hdfs://10.0.3.70:8020");
```

实现上传文件 upload 的方法：

```
/**
 *上传文件
 * @param localfile:本地的文件路径
 * @param remotefile:上传到 hdfs 上的文件路径
 * @throws Exception
 */
public void upload(String localfile,String remotefile) throws
Exception {
InputStream in=new BufferedInputStream(new
FileInputStream(localfile));
OutputStream out=fs.create(new Path(remotefile),new
```

```
Progressable( ) {
@ Override
public void progress( ) {
}
});
IOUtils. copyBytes( in,out,4096,true);
}
```

启动主函数，遍历/data/Webmagic 目录下的文件，把文件上传到 HDFS 的/user/PC/dict_in 目录。

```
public static void main(String[ ] args) throws Exception {
HdfsClient hdfsDB=new HdfsClient( );
File file=new File("/data/Webmagic");
if (file. exists( )) {
File[ ] files=file. listFiles( );
for (File file2 :files) {
if (!file2. isDirectory( )) {
hdfsDB. upload(file2. getAbsolutePath( ),"/user/PC/dict_in/" +file2. getName( ));
}
}
}
}
```

## 项目实训

**【实训任务】**

使用命令行与 Java 程序将采集的数据文件上传至 HDFS。

**【实训目的】**

1. 熟悉 HDFS 文件系统架构。
2. 学会 HDFS 相关属性配置方法。
3. 学会 HDFS 命令行上传操作。
4. 学会 Java 代码远程连接 HDFS 并上传文件的操作。

**【实训内容】**

将自己采集到的相关大数据的岗位信息，分别通过命令行方式与 Java 代码方式，上传至 HDFS 文件系统。

# 任务3.3　使用Sqoop转换MySQL中的学生成绩数据到Hive中

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习Sqoop的使用方法。

2. 学习将关系数据库（MySQL、Oracle）中的数据导入Hadoop的Hive中的方法。

3. 利用WebMagic爬取工具，将爬取的招聘岗位信息上传到Hive文件系统中。

## 任务目标

1. 熟悉Sqoop工具的使用方法。
2. 学会利用Navicat导入文件并创建表格的方法。
3. 学会利用Sqoop工具将MySQL表格导入Hive中的方法。

## 知识准备

Apache Sqoop是一个用于Apache Hadoop和关系数据库（RDBMS）之间进行数据传输的工具。通过Sqoop工具可以将数据从关系数据库（MySQL、Oracle等）中导入到Hadoop分布式文件系统中，在MapReduce中进行数据变换，将数据导出到关系数据库管理系统中。

Sqoop需要通过Hadoop的MapReduce从关系数据库中导入数据到HDFS中或者是Hive数据仓库中（实质上Hive中数据是存储在HDFS中的），使用ZooKeeper分布式协调工具来同步管理服务，Sqoop的应用部署是离不开Hadoop和ZooKeeper的。

## 任务实施

微课：3-3 使用Sqoop转换MySQL中的岗位数据到Hive中

（1）配置MySQL数据库

① 在Navicat中导入文件并创建表格。打开Navicat并连接上本地数据库，如图3-16所示。

在数据库recomm中导入文件。在连接列表中选择recomm数据库，在快捷工具栏中单击“导入向导”按钮，如图3-17所示。

② 在打开的窗口中选择文件格式，这里为txt文件。在“导入类型”选项组中选中“文字文件（*.txt）”单选按钮，最后单击“下一步”按钮，如图3-18所示。

③ 单击“导入从”文本框旁的浏览按钮，在打开的对话框中选择文件保存的路径。单击“编码”框右侧的“∨”按钮，在弹出的下拉列表中选择“65001（UTF-8）”选项，单击“下一步”按钮，如图3-19所示。

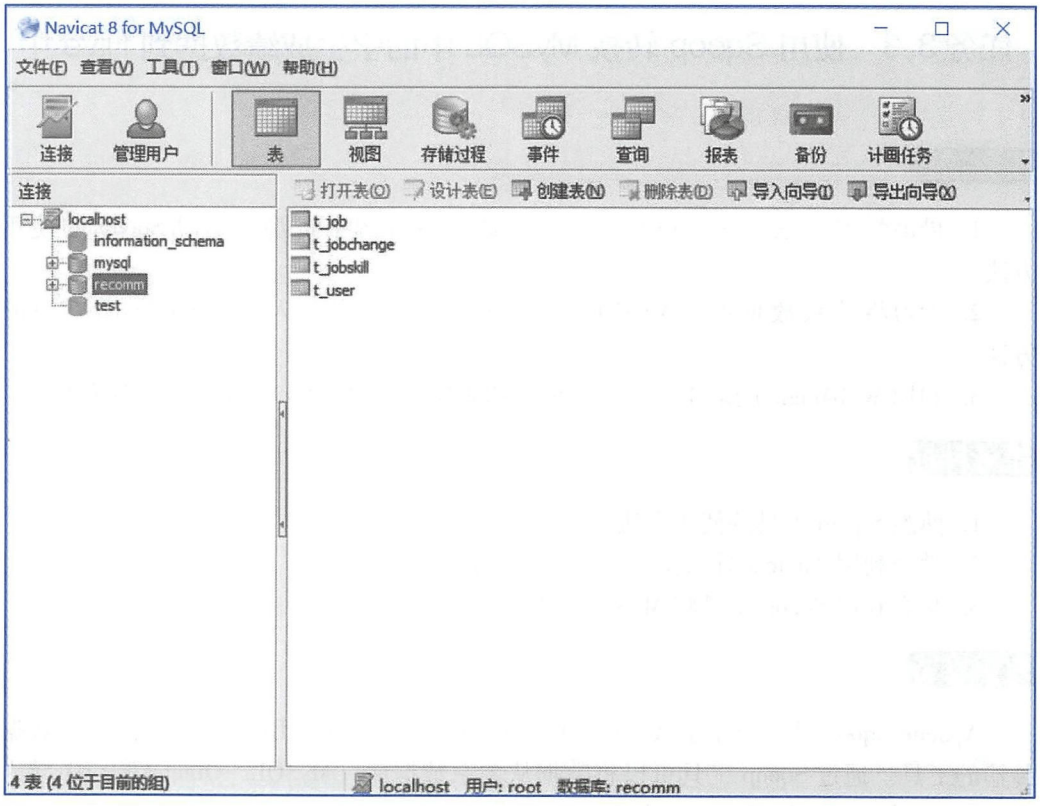



图 3-16 利用 Navicat 连接本地数据库

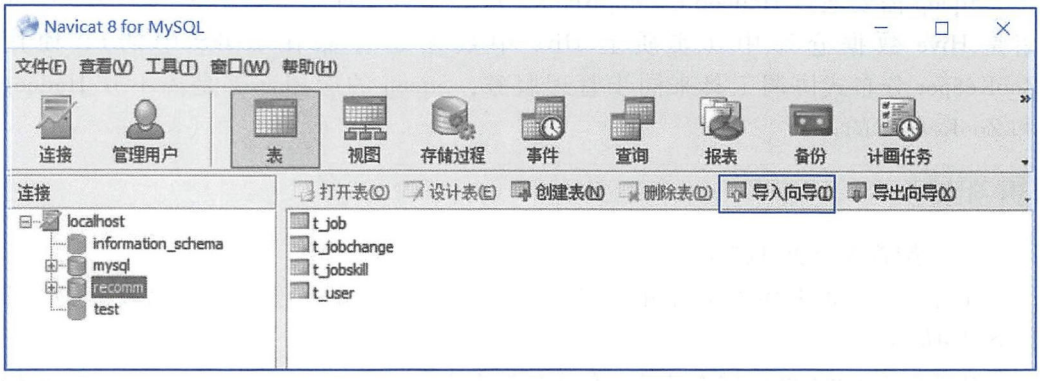



图 3-17 在数据库 recomm 中导入文件

④ 根据文件保存形式选择分隔符。选中“栏位定界符”选项组中的“逗号(,)”单选按钮，然后单击“下一步”按钮，如图 3-20 所示。

⑤ 默认情况下，系统会将导入文本文档的第 1 行作为字段名称。这里设置从文本文档的第 2 行开始导入，导入到数据库第一栏中。单击“下一步”按钮，如图 3-21 所示。

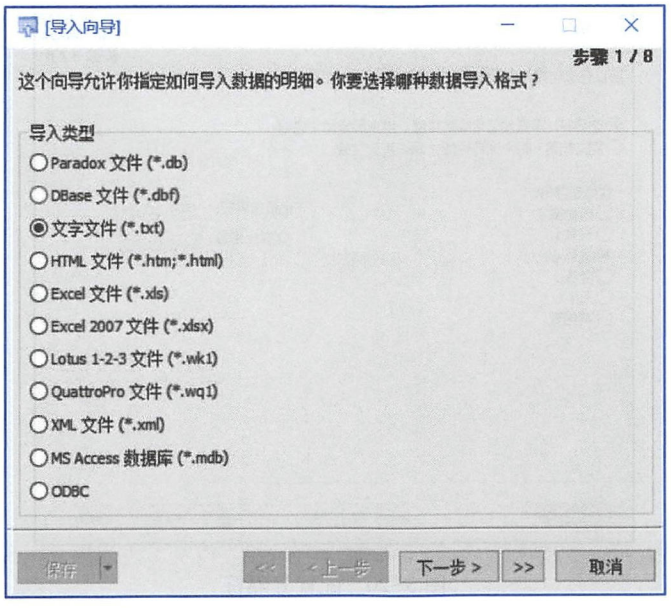



图 3-18　选择文件格式

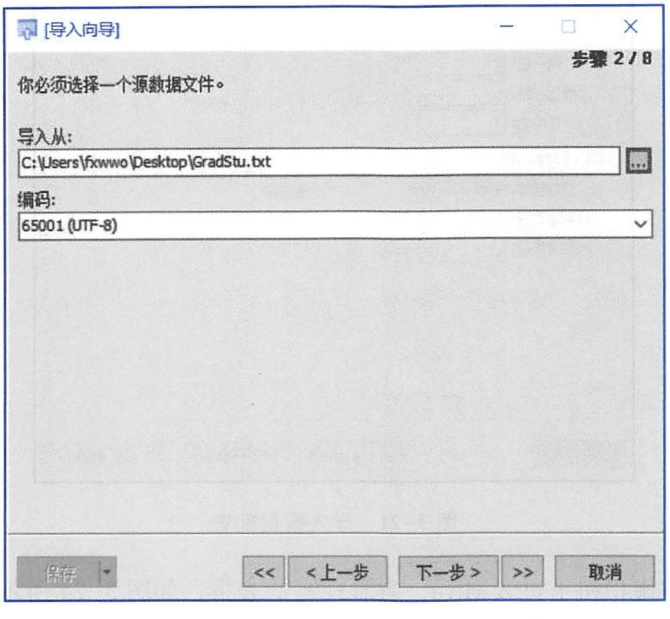



图 3-19　设置编码格式

⑥ 多次单击“下一步”按钮，使用默认配置，直到最后单击“开始”按钮，完成导入配置，如图 3-22 所示。

此时，该数据库下生成了名为 gradstu 的表格，打开后如图 3-23 所示。

（2）利用 Sqoop 将 MySQL 导入 Hive 中

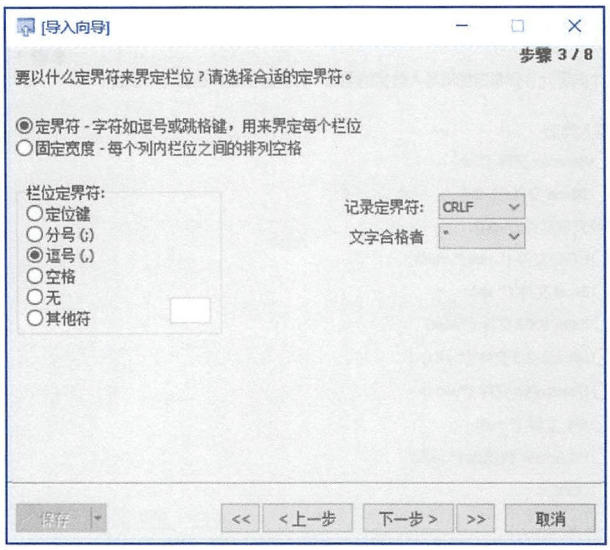



图 3-20 配置分隔符

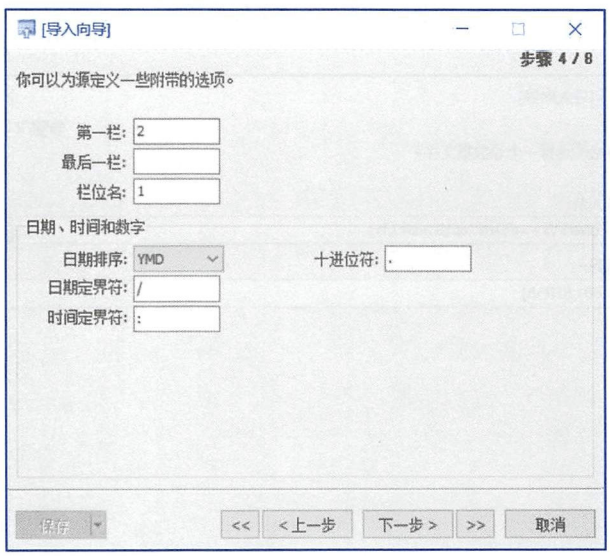



图 3-21 导入数据配置

① 在 Linux 虚拟机上进入 Hive，查看现有的表格，如图 3-24 所示。

② 退出 Hive，并使用“sqoop”命令行代码进行数据库导入，代码如下：

```
sqoop import --connect jdbc:mysql://192.168.0.134:3306/recomm --username root -P --table gradstu --hive-import -m 1
```

代码中的“192.168.0.134”即为数据库的 IP 地址，“3306”为端口，“root”为用户名，“-P”表示手动输入密码。

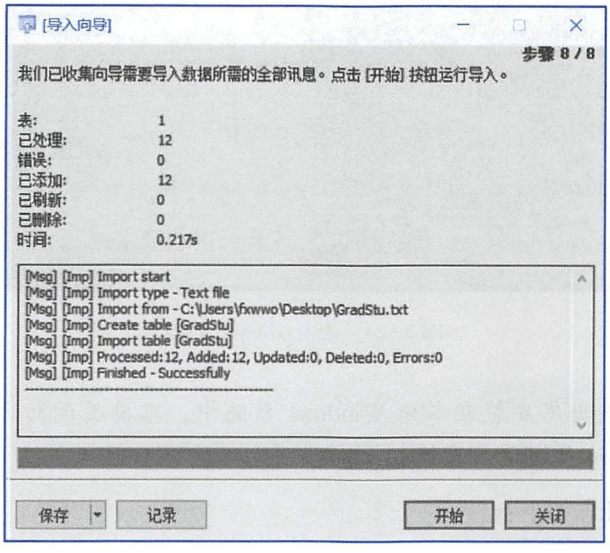
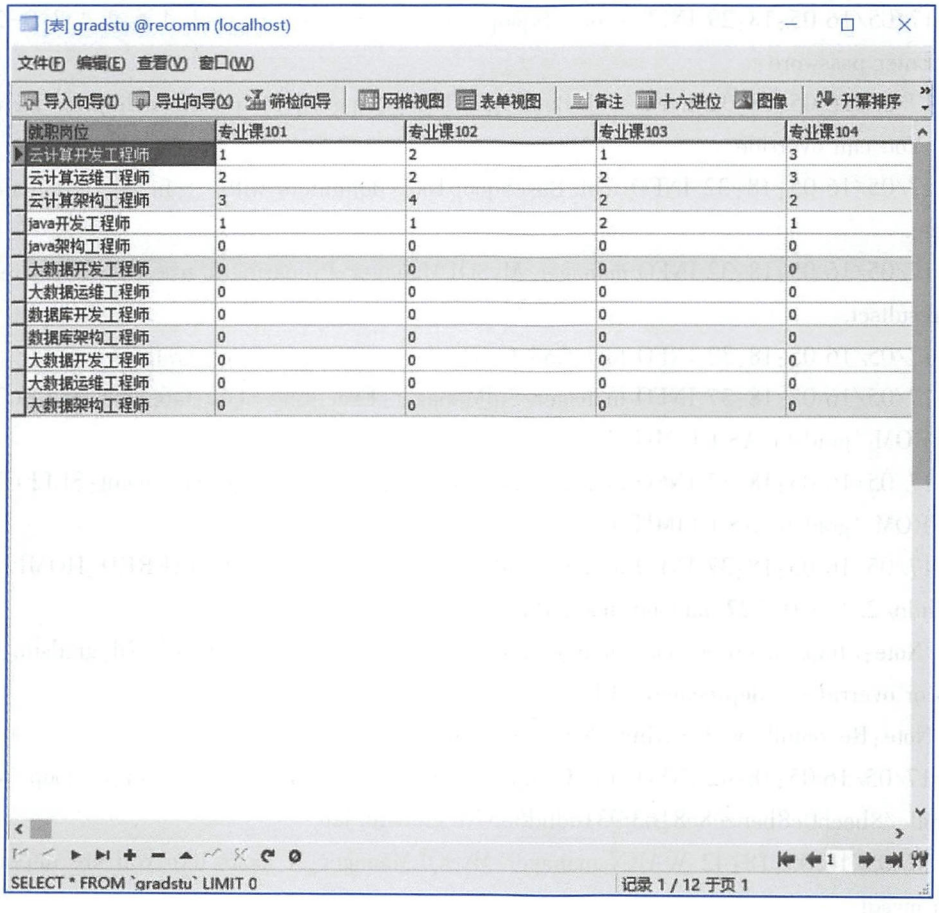



图 3-22　完成导入操作




| 就职岗位 | 专业课101 | 专业课102 | 专业课103 | 专业课104 |
|---|---|---|---|---|
| 云计算开发工程师 | 1 | 2 | 1 | 3 |
| 云计算运维工程师 | 2 | 2 | 2 | 3 |
| 云计算架构工程师 | 3 | 4 | 2 | 2 |
| java开发工程师 | 1 | 1 | 2 | 1 |
| java架构工程师 | 0 | 0 | 0 | 0 |
| 大数据开发工程师 | 0 | 0 | 0 | 0 |
| 大数据运维工程师 | 0 | 0 | 0 | 0 |
| 数据库开发工程师 | 0 | 0 | 0 | 0 |
| 数据库架构工程师 | 0 | 0 | 0 | 0 |
| 大数据开发工程师 | 0 | 0 | 0 | 0 |
| 大数据运维工程师 | 0 | 0 | 0 | 0 |
| 大数据架构工程师 | 0 | 0 | 0 | 0 |

图 3-23　查看导入结果

```
hive> show tables;
OK
apply_for_job
course
hbase_table1
ip_to_country
phy_opt_course
phy_opt_course_score
t_job
temp
weblog_entries
weblog_entries_url_length
Time taken: 2.213 seconds, Fetched: 10 row(s)
```

图 3-24 查看 Hive 中的表

**注意**：如果数据库安装在本地 Windows 系统中，需要关闭防火墙，才能使远程 Linux 虚拟机与本地 Windows 系统相互 ping 通。

代码运行的日志如下：

```
Warning:/usr/hdp/2.4.3.0-227/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
17/05/16 05:18:29 INFO sqoop.Sqoop:Running Sqoop version:1.4.6.2.4.3.0-227
Enter password:
17/05/16 05:18:32 INFO tool.BaseSqoopTool:Using Hive-specific delimiters for output. You can override
17/05/16 05:18:32 INFO tool.BaseSqoopTool:delimiters with --fields-terminated-by,etc.
17/05/16 05:18:32 INFO manager.MySQLManager:Preparing to use a MySQL streaming resultset.
17/05/16 05:18:32 INFO tool.CodeGenTool:Beginning code generation
17/05/16 05:18:37 INFO manager.SqlManager:Executing SQL statement:SELECT t.* FROM 'gradstu'AS t LIMIT 1
17/05/16 05:18:37 INFO manager.SqlManager:Executing SQL statement:SELECT t.* FROM 'gradstu'AS t LIMIT 1
17/05/16 05:18:37 INFO orm.CompilationManager:HADOOP_MAPRED_HOME is /usr/hdp/2.4.3.0-227/hadoop-mapreduce
Note:/tmp/sqoop-root/compile/8beeb0e8bebec8c8163f931edbd6837d/gradstu.Java uses or overrides a deprecated API.
Note:Recompile with -Xlint:deprecation for details.
17/05/16 05:18:42 INFO orm.CompilationManager:Writing jar file:/tmp/sqoop-root/compile/8beeb0e8bebec8c8163f931edbd6837d/gradstu.jar
17/05/16 05:18:42 WARN manager.MySQLManager:It looks like you are importing from mysql.
```

```
17/05/16 05:18:42 WARN manager. MySQLManager:This transfer can be faster! Use the --direct
17/05/16 05:18:42 WARN manager. MySQLManager:option to exercise a MySQL-specific fast path.
17/05/16 05:18:42 INFO manager. MySQLManager:Setting zero DATETIME behavior to convertToNull (mysql)
17/05/16 05:18:42 INFO mapreduce. ImportJobBase:Beginning import of gradstu
SLF4J:Class path contains multiple SLF4J bindings.
SLF4J:Found binding in [jar:file:/usr/hdp/2. 4. 3. 0-227/hadoop/lib/slf4j-log4j12-1. 7. 10. jar!/org/slf4j/impl/StaticLoggerBinder. class]
SLF4J:Found binding in [jar:file:/usr/hdp/2. 4. 3. 0-227/zookeeper/lib/slf4j-log4j12-1. 6. 1. jar!/org/slf4j/impl/StaticLoggerBinder. class]
SLF4J:See http://www. slf4j. org/codes. html#multiple_bindings for an explanation.
SLF4J:Actual binding is of type [org. slf4j. impl. Log4jLoggerFactory]
17/05/16 05:18:45 INFO impl. TimelineClientImpl:Timeline service address:http://slave1:8188/ws/v1/timeline/
17/05/16 05:18:45 INFO client. RMProxy:Connecting to ResourceManager at slave1/10. 0. 3. 71:8050
17/05/16 05:18:57 INFO db. DBInputFormat:Using read commited transaction isolation
17/05/16 05:18:57 INFO mapreduce. JobSubmitter:number of splits:1
17/05/16 05:18:58 INFO mapreduce. JobSubmitter:Submitting tokens for job:job_1494473183676_0015
17/05/16 05:18:58 INFO impl. YarnClientImpl:Submitted application application_1494473183676_0015
17/05/16 05:18:58 INFO mapreduce. Job:The url to track the job:http://slave1:8088/proxy/application_1494473183676_0015/
17/05/16 05:18:58 INFO mapreduce. Job:Running job:job_1494473183676_0015
17/05/16 05:19:22 INFO mapreduce. Job:Job job_1494473183676_0015 running in uber mode :false
17/05/16 05:19:22 INFO mapreduce. Job:  map 0% reduce 0%
17/05/16 05:19:39 INFO mapreduce. Job:  map 100% reduce 0%
17/05/16 05:19:44 INFO mapreduce. Job:Job job_1494473183676_0015 completed successfully
17/05/16 05:19:44 INFO mapreduce. Job:Counters:30
    File System Counters
        FILE:Number of bytes read=0
```

```
                FILE: Number of bytes written=152719
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=87
                HDFS: Number of bytes written=578
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=13484
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=13484
                Total vcore-seconds taken by all map tasks=13484
                Total megabyte-seconds taken by all map tasks=9196088
        Map-Reduce Framework
                Map input records=12
                Map output records=12
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=86
                CPU time spent (ms)=1390
                Physical memory (bytes) snapshot=121323520
                Virtual memory (bytes) snapshot=2451861504
                Total committed heap usage (bytes)=126681088
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=578
    17/05/16 05:19:44 INFO mapreduce.ImportJobBase: Transferred 578 bytes in
60.3772 seconds (9.5731 bytes/sec)
    17/05/16 05:19:44 INFO mapreduce.ImportJobBase: Retrieved 12 records.
    17/05/16 05:19:49 INFO manager.SqlManager: Executing SQL statement: SELECT t.
```

```
* FROM 'gradstu' AS t LIMIT 1
    17/05/16 05:19:49 INFO hive.HiveImport: Loading uploaded data into Hive Logging initialized using configuration in jar:file:/usr/hdp/2.4.3.0-227/hive/lib/hive-common-1.2.1000.2.4.3.0-227.jar!/hive-log4j.properties
    OK
    Time taken: 17.378 seconds
    Loading data to table default.gradstu
    Table default.gradstu stats: [numFiles=1, totalSize=578]
    OK
    Time taken: 1.735 seconds
```

运行成功后再进入 Hive，并查看所有表格，如图 3-25 所示。

```
hive> show tables
    > ;
OK
apply_for_job
course
gradstu
hbase_table1
ip_to_country
phy_opt_course
phy_opt_course_score
t_job
temp
weblog_entries
weblog_entries_url_length
Time taken: 2.292 seconds, Fetched: 11 row(s)
```

图 3-25　查看导入结果

可以看到，已经有了 gradstu 表格。现在查看该表格内容，运行如下代码：

```
hive> select * from gradstu;
```

发现表格已经导入，至此功能已实现，如图 3-26 所示。

```
hive> select * from gradstu
    > ;
OK
云计算开发工程师        1       2       1       3       0       0       0       0       0
云计算运维工程师        2       2       2       3       0       0       0       0       0
云计算架构工程师        3       4       2       2       0       0       0       0       0
java开发工程师  1       1       2       1       0       0       0       0       0
java架构工程师  0       0       0       0       1       2       1       3       0       0
大数据开发工程师        0       0       0       0       2       2       2       3       0
大数据运维工程师        0       0       0       0       3       4       2       2       0
数据库开发工程师        0       0       0       0       1       1       2       1       0
数据库架构工程师        0       0       0       0       0       0       0       0       3
大数据开发工程师        0       0       0       0       0       0       0       0       3
大数据运维工程师        0       0       0       0       0       0       0       0       2
大数据架构工程师        0       0       0       0       0       0       0       0       1
Time taken: 0.852 seconds, Fetched: 12 row(s)
hive>
```

图 3-26　查询表中数据

## 项目实训

**【实训任务】**

1. 使用 Sqoop 实现 MySQL 到 Hive 的数据导入。
2. 使用 Sqoop 实现 MySQL 到 HDFS 的数据导入。

**【实训目的】**

1. 熟悉 Sqoop 的主要功能。
2. 学会在 Windows/Linux 操作系统下，导入 MySQL 数据库/表的操作。
3. 学会使用 Sqoop 进行远程连接并传输数据表的操作。

**【实训内容】**

1. 将采集到的相关大数据的岗位信息，导入本地 MySQL 数据库，再通过 Sqoop 工具上传至 Hive 数据库。

2. 将采集到的相关大数据的岗位信息，导入本地 MySQL 数据库，再通过 Sqoop 工具上传至 HDFS 数据库。

## 项目小结

本项目主要讲解了数据采集的功能，其中分为网络爬虫与数据传输两部分。网络爬虫部分选用了 WebMagic 爬虫框架，其基于 Java 开发的特点与学情分析系统开发环境（Java EE）相吻合，同时通过网络爬虫的开发，帮助读者熟悉 HTML 的基本结构以及解析网页内容的方法。在数据传输部分，主要实现了本地文件系统与 HDFS 文件系统的连接，以及 MySQL 数据库与 Hive 数据仓库的数据传输。

# 项目 4

# 岗位数据的处理

PPT4：岗位数据的处理

## 学习目标

【知识目标】

- 掌握 MapReduce 计算框架。
- 了解 Hive 内部读写流程。

【技能目标】

- 熟悉 MapReduce Java API 代码规范。
- 学会 Hive 数据仓库基本操作命令。
- 学会 Java API 对 Hive 的读写操作。

## 学习情境

在将岗位信息抓取下来之后，米老师发现，这些文本信息并不能直接用于文本聚类。由于公司发布招聘信息的自主性，会导致岗位名称千差万别，一个岗位存在着数个岗位名称，这不利于后续文本聚类。因此米老师希望可以通过一定的方法，将岗位名称规范化，并在岗位描述中抽取其中关键的技能点（关键词）。他打算通过 MapReduce 计算框架，定制新的清洗规则，高效地对大量文本信息进行数据预处理。

同时，米老师通过对上传至 Hive 数据库的数据表进行数据预处理，以及必要的统计分析，为后续大数据分析做好准备工作。

# 任务 4.1 使用 Hadoop MR 对数据进行清洗

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习 MapReduce 的相关技术知识。

2. 编程实现对岗位描述的规范化操作。

3. 编程实现对岗位描述的分词操作。

4. 使用 Hadoop MR，完成数据清洗操作。

## 任务目标

1. 学会 MapReduce 框架定制操作。

2. 学会对岗位名称进行规范化操作。

3. 学会对岗位描述进行分词操作，建立技能点词库，匹配岗位描述中出现的技能点。

4. 学会项目程序的安装与部署操作。

## 知识准备

### 1. MapReduce

Hadoop MapReduce 是一个软件框架，基于该框架能够容易地编写应用程序，这些应用程序能够运行在由上千个商用机器组成的大集群上，并以一种可靠的、具有容错能力的方式并行地处理上 TB 级别的海量数据集。

笔记

### 2. MapReduce 的功能

简单地讲，MapReduce 可以做大数据处理。所谓大数据处理，即以价值为导向，对大数据进行加工、挖掘和优化等各种处理。

MapReduce 擅长处理大数据，这是由 MapReduce 的“分而治之”设计思想决定的。

（1）Mapper 负责“分”

即把复杂的任务分解为若干个“简单的任务”来处理。“简单的任务”包含以下 3 层含义：

① 数据或计算的规模相对原任务要大大缩小。

② 就近计算原则，即任务会分配到存放着所需数据的节点上进行计算。

③ 这些小任务可以并行计算，彼此间几乎没有依赖关系。

（2）Reducer 负责对 Map 阶段的结果进行汇总

至于需要多少个 Reducer，用户可以根据具体问题，通过在 mapred-site. xml 配置

文件里设置参数 mapred. reduce. tasks 的值来实现，默认值为 1。

具体的 MapReduce 流程如图 4-1 所示。

Map 端的任务是对每一个（<key/value>）键值对进行具体任务处理，而在由 Map 端传递到 Reduce 的过程中自动按 Key 值进行排序，并在 Reduce 端进行整合去重的工作。

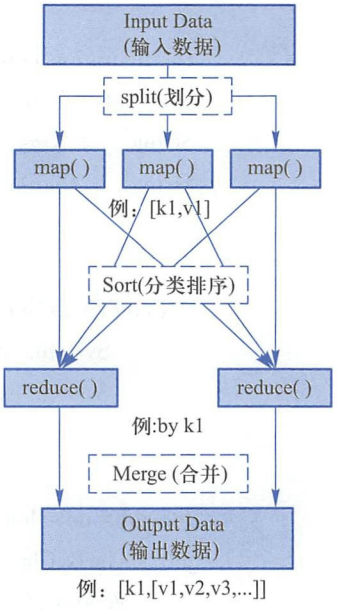



图 4-1　MapReduce 流程图

### 3. MapReduce 工作机制

MapReduce 的整个工作过程包含如下 4 个独立的实体。

① 客户端，用来提交 MapReduce 作业。

② JobTracker，用来协调作业的运行。

③ TaskTracker，用来处理作业划分后的任务。

④ HDFS，用来在其他实体间共享作业文件。

## 任务实施

（1）定制 MapReduce 框架

MapReduce 是一个计算框架，既然是做计算的框架，其表现形式就有个输入（input），MapReduce 操作该输入，通过本身定义好的计算模型，得到一个输出（output），该输出就是所需要的结果。

这里需要学习的就是这个计算模型的运行规则。在运行一个 MapReduce 计算任务时，任务过程被分为 Map 阶段和 Reduce 阶段两个阶段，每个阶段都是用键值对（key/value）作为输入（input）和输出（output）。而用户要做的就是定义好这两个阶段的函数：map 函数和 reduce 函数。

以下是本案例使用的 MapReduce 框架代码：

微课：4-1 使用 Hadoop MR 对数据进行清洗

代码 4.1：BigDataStudyAnalysis——项目 4

```
public class CleanJobMR {
    public static class CleanJob Mapper extends Mapper<Object,Text,Text,Text> {
        public void map(Object key,Text value,Context context)
                        throws IOException,InterruptedException {
        }
}
    public static class CleanJobReducer extends Reducer<Text,Text,Text,Text> {

        public void reduce(Text key,Iterable<Text> values,
                        Context context) throws IOException,InterruptedException {
        }
```

```
    }
    public static void main(String[ ] args) throws Exception {

            Configuration conf=new Configuration( );
            conf. set( "fs. defaultFS" ,"hdfs://10. 0. 3. 70:8020" );
            String[ ] ioArgs=args;
            String[ ] otherArgs=new GenericOptionsParser( conf,ioArgs)
                        . getRemainingArgs( );

            if (otherArgs. length != 2) {
                    System. err. println( "Usage:Data Deduplication <in> <out>" );
                    System. exit(2);
            }

            Job job=new Job( conf,"Data Deduplication" );
            job. setJarByClass( CleanJobMR. class);
            // 设置 Map、Combine 和 Reduce 处理类
            job. setMapperClass( CleanJobMapper. class);
            job. setCombinerClass( CleanJobReducer. class);
            job. setReducerClass( CleanJobReducer. class);
            // 设置输出类型
            job. setOutputKeyClass( Text. class);
            job. setOutputValueClass( Text. class);
            job. setOutputFormatClass( SequenceFileOutputFormat. class);
            // 设置输入和输出目录
            FileInputFormat. addInputPath( job,new Path( otherArgs[0]));
            FileOutputFormat. setOutputPath( job,new Path( otherArgs[1]));
            System. exit( job. waitForCompletion( true) ? 0 :1);
        }
    }
```

以下对该框架结构进行详细的解析。

首先，是继承类 CleanJobMapper。

```
public static class CleanJobMapper extends   Mapper<Object,Text,Text,Text> {…}
```

其中<Object,Text,Text,Text>代表的是输入 key/value 对和输出 key/value 对的格式。继承类 CleanJobReducer 也是一样的道理。

在 CleanJobMapper 中的核心方法就是 map 方法：

非卖品，仅供非商业用途或交流学习使用

```
public void map(Object key,Text value,Context context)
                    throws IOException,InterruptedException {
    }
```

其中有3个参数，前面2个Object key、Text value就是输入的key和value，第3个参数Context context可以记录输入的key和value，如context.write（word，one）。此外context还会记录map运算的状态。

在CleanJobReducer中的核心方法就是reduce方法：

```
public void reduce(Text key,Iterable<Text> values,
                    Context context) throws IOException,InterruptedException {
    }
```

reduce函数的输入也是一个key/value的形式，不过它的value是一个迭代器的形式Iterable<Text> values，即reduce的输入是一个key对应一组value值，reduce也有context，其作用与map的context一致。

在MapReduce的处理过程中，由于MapReduce会在Map~Reduce中将重复的Key合并在一起，所以MapReduce很容易就去除重复的行。

Map无须做任何处理，设置Map中写入context的东西为不作任何处理的行，也就是Map中最初处理的value即可；而Reduce同样无须做任何处理，写入输出文件的内容就是最初得到的Key。

具体的定制map方法以及reduce方法要结合数据清洗的逻辑来编写，其将在后续步骤中说明。

此外，还有一个重要的main函数的调用，需要详细讲述。首先是：

```
Configuration conf=new Configuration();
conf.set("fs.defaultFS","hdfs://10.0.3.70:8020");
```

运行MapReduce程序前都要初始化Configuration，该类主要是读取HDFS和MapReduce系统配置信息，这些配置信息可以在Hadoop安装路径的conf下找到或者在Ambari图形界面中“HDFS”→“配置”→“Advanced”中找到。所涉及的文件有core-site.xml、hdfs-site.xml和mapred-site.xml等。程序员开发MapReduce时只是在做填空，在map函数和reduce函数中编写实际进行的业务逻辑，其他的工作都交给MapReduce框架自己操作。

接着看如下代码：

```
String[] ioArgs=args;
        String[] otherArgs=new GenericOptionsParser(conf,ioArgs)
                        .getRemainingArgs();
        if (otherArgs.length != 2) {
                System.err.println("Usage:Data Deduplication <in> <out>");
```

非卖品，仅供非商业用途或交流学习使用

```
            System.exit(2);
        }
```

需要注意的是，在运行数据清洗 CleanJobMR 程序时需要两个参数，否则就会报错退出。至于 GenericOptionsParser 类，是用来解释常用的 Hadoop 命令的，并根据需要为 Configuration 对象设置相应的值，在平常的开发中很少用到它，一般是让类实现 Tool 接口，然后在 main 函数中使用 ToolRunner 运行程序，而 ToolRunner 内部会调用该 GenericOptionsParser 类。

接着看如下代码：

```
Job job=new Job(conf,"Data Deduplication");
    job.setJarByClass(CleanJobMR.class);
    job.setMapperClass(CleanJobMapper.class);
    job.setCombinerClass(CleanJobReducer.class);
    job.setReducerClass(CleanJobReducer.class);
```

第 1 行是在构建一个 job，在 MapReduce 框架中，1 个 MapReduce 任务既可以叫作 MapReduce 作业，也可以叫作 1 个 MapReduce 的 job，而具体的 map 和 reduce 运算就是 task。这里构建 job 时，需要有 2 个参数，分别是 conf 和 job 的名称。

第 2 行是装载编写好的计算程序，假设程序类名是 CleanJobMR。需要强调得是，虽然编写 MapReduce 程序只需要实现 map 函数和 reduce 函数，但是实际开发要实现 3 个类，第 3 个类是为了配置 MapReduce 如何运行 map 和 reduce 函数。准确地说，就是构建 1 个 MapReduce 能执行的 job，如 CleanJobMR 类。

第 3 行和第 5 行是装载 map 函数和 reduce 函数的实现类，第 4 行是装载 Combiner 类，实际上本案例去掉第 4 行也没有关系，但理论上使用了第 4 行其运行效率会更高。

接着看如下代码：

```
job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    job.setOutputFormatClass(SequenceFileOutputFormat.class);
```

其定义输出 key/value 的类型，即最终存储在 HDFS 上结果文件的 key/value 的类型，以及定义保存文件的格式，这里为序列文件。

接着看如下代码：

```
FileInputFormat.addInputPath(job,new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job,new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 :1);
```

第 1 行就是构建输入的数据文件，第 2 行是构建输出的数据文件，最后一行如果 job 运行成功了，程序就会正常退出。

非卖品，仅供非商业用途或交流学习使用

项目 MapReduce 的简略图如图 4-2 所示，这里将岗位信息（名称+描述）作为 Key 值，在 Map 端进行规范化处理，然后在 Reduce 端根据 Key 值（即岗位信息）进行排序整合与去重工作。

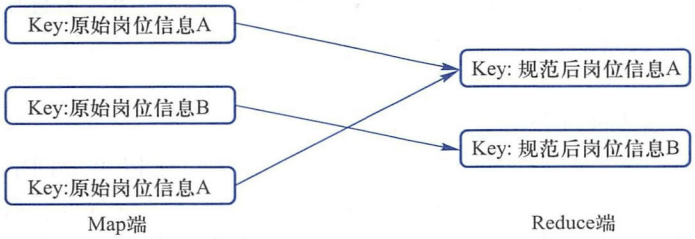



图 4-2　MapReduce 的简略图

（2）规范化岗位名称

由于招聘网站上的招聘信息编辑工作是由各个公司自主完成发布的，因此存在着数据信息的多样性，即使在抓取环节过滤掉了大量的“垃圾”岗位信息，依然存在各种各样的岗位名称表述方式。如图 4-3 所示，框中的岗位名称，其实都可以宽泛地统称为“云计算开发工程师”（不分初中高等级），而由于各个公司不同的命名规则，导致出现了 7 种不同的岗位名称，这非常不利于在后续数据分析环节进行文本聚类，因此需要对岗位名称进行规范化。

| 职位名 | 公司名 | 工作地点 | 薪资 |
| --- | --- | --- | --- |
| 高级云计算研发工程师 | 南京优速网络科技有限公司 | 南京-江宁区 | 17-25万/ |
| 云计算开发高级工程师 | 深圳市科思科技股份有限公司 | 深圳 | 2-3万/ |
| 云计算及服务器存储技术支持工程师 | 广东新源信息技术有限公司 | 广州-天河区 | 0.6-1.1万 |
| 大数据及云计算产品经理 | 广州清宇信息科技有限公司 | 广州 | 20-30万/ |
| 云计算开发初级工程师（无经验可实习... | 众阮科技（上海） | 上海 | 4.5-6千/ |
| 云计算开发工程师 | 广东铭鸿数据有限公司 | 广州-海珠区 | 12-20万/ |
| 云计算解决方案咨询师 | 上海司羽通信设备有限公司 | 北京 | 1.5-2.5万 |
| 云项目云计算研发高级工程师 | 旭孚（北京）新能源科技有限公司 | 北京-大兴区 | 20-30万/ |
| 售前工程师（云计算方向） | 南凌科技股份有限公司 | 上海-徐汇区 | 1-1.5万/ |
| 大数据云计算 | 广州商脉网络科技有限公司 | 广州-天河区 | 1-1.5万/ |
| 云计算软件工程师 | 西安知象光电科技有限公司 | 西安-雁塔区 | 20-30万/ |
| 云计算架构师 | 湖南天云软件技术有限公司 | 长沙 | 0.8-1.5万 |

图 4-3　招聘信息

案例中将所有的岗位名称规范为“××（名词）+××（动词）+工程师”的形式。

非卖品，仅供非商业用途或交流学习使用

笔记

其中的名词部分为云计算、大数据、数据库等关键词，动词为开发、运维、测试、集成、实施等，而最后一个称谓设置为工程师。在本案例中只分析技术类的岗位，在技术类的岗位描述中包含有更多的技能点，这有利于后续进行文本聚类。

对于岗位规范化动词部分的提取，则相对简单，即判断岗位名称中是否含有词库中设定的动词且只含有一个，若没有匹配到动词，则默认为“开发”。

岗位名称规范化的代码由于并不唯一，具体可以见工程。

（3）对岗位描述进行分词

由于在案例中需要提取岗位对应的技能点，技能点是以单个词汇的形式出现的（如 Java、C++、Linux 等），所以案例中需要对岗位描述进行分词。有的用户或许会认为没有必要进行分词，直接用一个 contains 语句判断岗位描述中是否含有某个技能点词汇即可。该解决方案在大部分情况下是可行的，但存在一定的弊端。例如，一个岗位描述中出现了 JavaScript 这个技能点，对其不进行分词而直接通过 contains 语句进行操作的话，将匹配出 Java 以及 JavaScript 两个技能点。由于本案例着重于技能点和岗位名称的文本聚类，因此需要匹配技能点的操作，这有利于进行更广泛的文本聚类。因此学会分词处理，也是为进一步的学习文本聚类打下基础。

① 需要选择合适的分词器，本案例中选用 Lucene 4.6.1 的分词类，代码如下：

```
Analyzer analyzer=new StandardAnalyzer(Version.LUCENE_46);
```

② 选择需要分词的岗位描述，本案例根据前面数据的保存格式，应该是在每一行的第 10 个位置：

```
String str=value.toString();
//简单 split,原因是前面抓取的规则就是\t 来间隔
String[] strs=str.split("\t");
//对第十个内容分词
String dictStr=strs[9];
String temp="";
StringReader reader=new StringReader(dictStr);
//循环累加出所有的分词
TokenStream ts=analyzer.tokenStream(dictStr,reader);
ts.reset();
boolean hasnext= ts.incrementToken();
String[] tempdict= Dictionary.dictionary1;
while (hasnext) {
  //技能关键词过滤,在字典里面有的才加入
  CharTermAttribute ta=ts.getAttribute(CharTermAttribute.class);
  boolean flag=ArrayUtils.contains(tempdict,ta.toString().toLo werCase());
  if (flag)
  {
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
            temp += ta.toString() + "\t";
            List<String> templist=new ArrayList<String>(Arrays.asList(tempdict));

            templist.remove(ta.toString().toLowerCase());
            tempdict=templist.toArray(new String[0]);
            count++;
        }
        hasnext=ts.incrementToken();
    }
    reader.close();
    ts.end();
    ts.close();
```

## 项目实训

**【实训任务】**

利用 MapReduce 计算框架实现数据预处理操作。

**【实训目的】**

1. 熟悉 MapReduce 计算框架。
2. 熟悉 MapReduce 的 Java API 代码。
3. 学会利用 Map 端定制数据清洗规则，利用 Reduce 端进行数据去重排序操作。

**【实训内容】**

自行定制所需的清洗规则，如规范化薪资数据，将地区信息转换为所在省份信息等，利用 MapReduce 程序进行数据预处理。

# 任务 4.2　使用命令行对 Hive 进行数据查询和过滤

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习 Hive 的相关技术知识。
2. 使用命令行，完成对 Hive 的数据查询和过滤操作。

## 任务目标

1. 学会使用命令行过滤不符合应用需求的数据操作方法。
2. 学会使用命令行查询符合条件的数据并展示的操作方法。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



## 知识准备

笔记

Hive 是 Hadoop 项目中一个重要的子项目，也是 Apache 项目基金会下的一个开源项目。Apache Hive 将数据存储在 HDFS 文件系统中，并在该文件系统中进行大数据分析。它提供了类 SQL 的编程接口，可以将结构化的数据文件映射为一张数据库表，实现了部分 SQL 语句，并可以将 SQL 语句转换为 MapReduce 任务进行运行，从而快速实现简单的 MapReduce 统计，不需要开发专门的 MapReduce 应用。

Hive 数据逻辑上由存储的数据和描述表格中的数据形式的相关元数据组成。由于元数据可能要面临不断地更新、修改和读取，Hive 中的数据存放在 HDFS 文件系统中，元数据显然不适合用 Hadoop 文件系统进行存储，其元数据是存储在 RDBMS（关系型数据库管理系统）中，用户在进行查询时，首先是到 RDBMS 中的读取元数据信息，从而也减少了语义检查的时间，提高了数据读取的效率。

同时在 Hadoop 生态系统中，Hive 允许用户使用 UDF（user defined function，用户定义函数）对数据进行处理，包括操作日期、字符串和其他数据挖掘工具。Apache Hive 使用了 DEFLATE、BWT、Snappy 数据算法对中间数据或最终数据进行数据压缩，从而提高数据吞吐量和性能。对数据做压缩，可以大量减少磁盘的存储空间，如基于文本的数据文件，可以将文件压缩 40%或更多，同时压缩后的文件在磁盘间传输和 I/O的时间也会大大减少。压缩和解压缩会带来额外的 CPU 开销，但却可以节省更多的 I/O 和使用更少的内存开销。Hadoop Jobs 作业，往往是 I/O 密集型的，而非 CPU 密集型的。数据压缩对 I/O 密集型的作业能大大提升性能，但是如果用户的 Jobs 作业是 CPU 密集型的，那么再使用压缩就会降低性能，这就需要用户对作业的类型做出判断，权衡是否要对数据做压缩。

Hive 定义了类 SQL 的语言 HiveQL，它允许用户进行和 SQL 相似的操作，这样开发人员能够方便地使用 Mapper 和 Reducer 操作，这对 MapReduce 框架是一个强有力的支持。虽然 HiveQL 与大部分的 SQL 语法相兼容，但它并不严格遵循支持 SQL-92 标准，如多表查询、支持 create table as select 和集成 MapReduce 脚本等。

## 任务实施

微课：4-2 用命令行对 Hive 进行数据查询和过滤

① 可以通过 Hive 命令粗略查看 Hive 仓库中的数据。

```
hive> select * from score;
```

返回结果：

```
OK
1       usera   90      90      80
2       userb   80      70      90
3       userc   70      70      60
4       userd   60      90      80
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
5                80          70          70
Time taken:0.281 seconds,Fetched:5 row(s)
```

② 需要将学生名字为空的错误数据过滤掉。首先将空字符串格式化为 NULL。

```
hive> alter table score set serdeproperties('serialization.null.format'='');
```

使用“select”命令再次查看格式化后数据的结果。

```
hive> select * from score;
```

返回结果：

```
OK
1        usera    90        90        80
2        userb    80        70        90
3        userc    70        70        60
4        userd    60        90        80
5        NULL     80        70        70
Time taken: 0.111 seconds,Fetched: 5 row(s)
```

接下来可以删除 username 为 NULL 的记录。

```
hive> insert overwrite table score select * from score where username is not null;
```

需要等待执行过程，如图 4-4 所示。

```
hive> insert overwrite table score select * from score where username is not null;
Query ID = root_20170511050637_082b0985-0933-41a4-ba25-0d778ac3b341
Total jobs = 1
Launching Job 1 out of 1


Status: Running (Executing on YARN cluster with App id application_1494473183676_0004)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 9.79 s
--------------------------------------------------------------------------------
Loading data to table stu.score
Table stu.score stats: [numFiles=1, numRows=4, totalSize=68, rawDataSize=64]
OK
Time taken: 12.917 seconds
hive>
```

图 4-4　删除 username 为 NULL 的记录

执行完成后再次使用“select”命令查看数据。

```
hive> select * from score;
```

返回结果：

```
OK
1        usera    90        90        80
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
2       userb   80      70      90
3       userc   70      70      60
4       userd   60      90      80
Time taken: 0.102 seconds, Fetched: 4 row(s)
```

可以看到 username 为 NULL 的记录已经被清除。

③ 使用“hive”命令查询并打印出所有成绩大于且等于 80 分的学生成绩记录。

```
hive> select * from score where score1>=80 and score2>=80 and score3>=80;
```

返回结果：

```
OK
1       usera   90      90      80
Time taken: 0.549 seconds, Fetched: 1 row(s)
```

### 项目实训

**【实训任务】**

利用 Hive 进行数据查询与数据表重写。

**【实训目的】**

1. 熟悉 Hive 基本操作命令。
2. 学会使用 Hive 添加、查询、重写数据表的基本操作。

**【实训内容】**

1. 使用 Hive 实现数据表的添加、查询和重写。
2. 查询在项目 3 中上传至 Hive 的数据表，进而进行相关过滤查询与重写操作。

## 任务 4.3 使用 Java API 对 Hive 进行连接与操作

### 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习使用 Java API 对 Hive 进行连接，并进行远程操作 Hive 数据的相关技术知识。

2. 使用 Java API，实现对 Hive 的连接与操作。

### 任务目标

1. 学会利用 Java 远程连接 Hive 的操作。
2. 学会对 Hive 中的数据，根据所在地统计省份信息操作。
3. 学会对 Hive 中的数据，实现岗位变化趋势的统计操作。
4. 学会对 Hive 中的数据，根据学历对薪资进行统计操作。

非卖品，仅供非商业用途或交流学习使用 严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



## 知识准备

Hive 提供了轻客户端的实现，通过 HiveServer 或者 HiveServer2，客户端可以在不启动 CLI 的情况下对 Hive 中的数据进行操作，两者都允许远程客户端使用多种编程语言如 Java、Python 向 Hive 提交请求，取回结果。HiveServer 或者 HiveServer2 都是基于 Thrift 的，但 HiveSever 有时被称为 ThriftServer，而 HiveServer2 却不会。这是因为 HiveServer 不能处理多于一个客户端的并发请求，其受 HiveServer 使用的 Thrift 接口所导致的限制，且无法通过修改 HiveServer 的代码进行修正。因此在 Hive-0. 11. 0 版本中重写 HiveServer 代码得到 HiveServer2，解决了该问题。HiveServer2 支持多客户端的并发和认证，为开放 API 的客户端（如 JDBC、ODBC）提供了更好的支持。

## 任务实施

微课：4-3 使用 Java API 对 Hive 进行连接与操作

（1）利用 Java 连接 Hive

① 在 Linux 命令行启动 Hiveserver2，在命令行窗口输入命令“hive --service hiveserver2”，如图 4-5 所示。或者直接输入命令“hiveserver2”启动。

```
[root@master ~]# hive --service hiveserver2
```

图 4-5　启动 HiveServer2

② 在 Java 中连接 Hive，代码如下：

```
import java. sql. Connection;
import java. sql. DriverManager;

public class Test {
    public static String driverName = "org. apache. hive. jdbc. HiveDriver";
    public static String url = "jdbc:hive2://192. 168. 0. 99:10000";
    public static void main(String[] args) throws Exception {
            Connection con;
            try {
                Class. forName(driverName);
                con = DriverManager. getConnection(url);
        System. out . println("数据库连接成功!");
            } catch (Exception e) {
                e. printStackTrace();
            }
    }
}
```

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



（2）根据所在地统计省份信息

① 在招聘网站上抓取到的岗位招聘信息中，地区信息表述呈现多样性，没有统一的要求。有的地区信息表述较粗，直接写到所在的省份，有的则表述较为具体，如具体到相应的城市和区、县等，见表 4-1。

表 4-1 岗位招聘信息

| 地区信息 | 地区信息 |
| --- | --- |
| 上海-浦东新区 | 成都 |
| 深圳 | 北京-海淀区 |
| 无锡 | 武汉-武昌区 |
| 上海 | 上海 |
| 佛山 | 天津-宁河县 |
| 上海-徐汇区 | 郑州 |
| 南京 | 上海 |
| 桂林 | 北京-海淀区 |
| 郑州 | 上海-浦东新区 |

要将这些地区信息统一转换为所在省份信息，看似是一个烦琐的工作，但其实只需要有两张数据表即可，一张为省份以及对应的省份 ID 的表，一张为城市以及对应省份 ID 的表。两张表结构见表 4-2 和表 4-3。

表 4-2 省份与省份 ID 信息表

| ID | Name | ID | Name |
| --- | --- | --- | --- |
| 1 | 北京 | 13 | 福建 |
| 2 | 天津 | 14 | 江西 |
| 3 | 河北 | 15 | 山东 |
| 4 | 山西 | 16 | 河南 |
| 5 | 内蒙古 | 17 | 湖北 |
| 6 | 辽宁 | 18 | 湖南 |
| 7 | 吉林 | 19 | 广东 |
| 8 | 黑龙江 | 20 | 广西 |
| 9 | 上海 | 21 | 海南 |
| 10 | 江苏 | 22 | 重庆 |
| 11 | 浙江 | 23 | 四川 |
| 12 | 安徽 | 24 | 贵州 |

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



续表

| ID | Name | ID | Name |
|---|---|---|---|
| 25 | 云南 | 30 | 宁夏 |
| 26 | 西藏 | 31 | 新疆 |
| 27 | 陕西 | 32 | 香港 |
| 28 | 甘肃 | 33 | 澳门 |
| 29 | 青海 | 34 | 台湾 |

表 4-3　城市与省份 ID 信息表

| Name | Province ID | Name | Province ID |
|---|---|---|---|
| 北京 | 1 | 衡水 | 3 |
| 天津 | 2 | 太原 | 4 |
| 石家庄 | 3 | 大同 | 4 |
| 唐山 | 3 | 阳泉 | 4 |
| 秦皇岛 | 3 | 长治 | 4 |
| 邯郸 | 3 | 晋城 | 4 |
| 邢台 | 3 | 朔州 | 4 |
| 保定 | 3 | 晋中 | 4 |
| 张家口 | 3 | 运城 | 4 |
| 承德 | 3 | 忻州 | 4 |
| 沧州 | 3 | 临汾 | 4 |
| 廊坊 | 3 | 吕梁 | 4 |

② 通过这两张表的关联，以及对地区文本信息的规范化，可以对 Hive 进行查询操作来统计省份信息。Hive 的查询语句与 SQL 一样，具体代码如下：

```
//查询地区信息,统一转换为省份信息
String sql = "select location from t_alljob";
ResultSet rs = stmt.executeQuery(sql);
while (rs.next()) {
    String tmp = rs.getString(1);
    if (tmp.contains("-")) {
        String[] strs = tmp.split("-");
        cityRec.add(strs[0]);
    }else if (tmp.contains("省")) {
```

严禁网络传播本PDF，违者责任自负！

版权所有，严禁传播，违者自负法律责任！



```
            cityRec.add(tmp.substring(0,tmp.length() - 1));
        }else {
            cityRec.add(tmp);
        }
    }
    Iterator<String> it=cityRec.iterator();
    while (it.hasNext()) {
        String cityTemp=it.next();
        String citySql="select name,provinceid from t_city where name=?";
        PreparedStatement cityStmt=con.prepareStatement(citySql);
        cityStmt.setString(1,cityTemp);
        ResultSet cityRs=cityStmt.executeQuery();
        if (cityRs.next()) {
            int provID=cityRs.getInt(2);
            String findProvSql="select name from t_province where id=?";
            PreparedStatement findProvStmt=con.prepareStatement(findProvSql);
            findProvStmt.setInt(1,provID);
            ResultSet findProvRs=findProvStmt.executeQuery();
            if (findProvRs.next()) {
                provRec.add(findProvRs.getString(1));
            }
        }else {
            String provSql="select name from t_province where name=?";
            PreparedStatement provStmt=con.prepareStatement(provSql);
            provStmt.setString(1,cityTemp);
            ResultSet provRs=provStmt.executeQuery();
            if (provRs.next()) {
                provRec.add(provRs.getString(1));
            }
        }
    }
    //省份计数
    Set<String> uniqueSet=new HashSet<String>(provRec);
    for (String temp : uniqueSet) {
        int tempCount=Collections.frequency(provRec,temp);
        System.out.println(temp+": "+tempCount+"个");
    }
```

严禁网络传播本PDF，违者责任自负！

③ 省份计数使用的是 HashSet。控制台输出结果如下：

```
数据库连接成功!
    青海:5 个          四川:525 个          贵州:144 个
    辽宁:200 个        内蒙古:37 个         天津:55 个
    新疆:22 个         江西:64 个           广西:61 个
    重庆:175 个        浙江:702 个          山西:36 个
    吉林:72 个         黑龙江:52 个         河北:40 个
    海南:20 个         西藏: 7 个           河南:109 个
    云南:80 个         宁夏:10 个           陕西:179 个
    福建:149 个        甘肃:8 个            广东:2255 个
    北京:2106 个       湖北:504 个          山东:223 个
    上海:1402 个       湖南:222 个
    江苏:835 个        安徽:179 个
```

（3）实现岗位变化趋势的统计

在岗位信息数据表 t_alljob 中，岗位的发布日期字段名为 date，见表 4-4。

表 4-4　岗位信息数据表

| date |
| --- |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |
| 05-06 发布 |

严禁网络传播本PDF，违者责任自负！

因此仅需要通过简单的查询以及利用 HashSet 进行技术统计即可，具体代码如下：

```
//分析每日变化量
        String selectSql="select date from t_alljob";
        PreparedStatement stmt1=con.prepareStatement(selectSql);
        ResultSet rs=stmt1.executeQuery();
        while(rs.next()){
            dateList.add(rs.getString(1));
        }
//计数并打印至控制台
        Set<String> dateSet=new HashSet<String>(dateList);
        for(String temp: dateSet){
          String realDate=temp.substring(0,temp.length()-2);
          int count=Collections.frequency(dateList,temp);
          System.out.println(realDate+":"+count+"个");
        }
```

输出台结果如下：

```
数据库连接成功!
    06-04:100 个        05-25:43 个         05-13:18 个
    05-20:86 个         05-12:296 个        05-27:190 个
    06-05:146 个        05-22:141 个        05-14:122 个
    05-21:115 个        06-06:131 个        05-28:97 个
    06-02:57 个         05-23:117 个        0507:242 个
    05-31:242 个        05-10:203 个        06-01:119 个
    05-09:760 个        05-15:254 个        05-08:3605 个
    06-03:5 个          05-16:253 个        05-18:34 个
    05-24:64 个         05-17:106 个        05-19:169 个
    05-11:148 个        05-26:146 个        05-06:2503 个
```

这里 05-06 与 05-08 所发布数量显著大于其他日期，这是因为起始日期为 05-06，因此每一个岗位都看作是当天新发布的，之后出现许多重复岗位只是更新了日期，因此通过设置网页 URL 为主键即可过滤重复信息。

（4）根据学历对薪资进行统计

① 招聘网站上的薪资信息也没有统一的格式，需要使用程序来人为地进行规范化处理，原始的数据见表 4-5。

严禁网络传播本PDF，违者责任自负！

表 4-5　薪资信息原始数据

| salary | |
|---|---|
| 0.7~1.5 万/月 | 1~1.5 万/月 |
| 0.8~1 万/月 | 4~6 千/月 |
| 6~8 千/月 | 2.5~3 万/月 |
| 2~3 万/月 | 5~8 千/月 |
| 4.5~6 千/月 | 3.5~6.5 千/月 |
| 1.5~2 万/月 | 6~8 千/月 |
| 0.6~1.2 万/月 | 1~1.5 万/月 |
| 3~4.5 千/月 | 0.7~1 万/月 |
| 2~3 千/月 | 3.5~7 千/月 |
| 6~8 千/月 | |

可以看出，有以年（月）为时间度量的，有以千（万）为货币度量的，理论上便存在 4 种组合形式，同时还要排除不以（数字-数字）出现的形式，因此具体参考如下代码实现。

```
//规范化月薪
        if (jobSalary.contains("年") && jobSalary.contains("-")){
            choose1=true ;
            if (jobSalary.contains("万")){
              String[ ] s1=jobSalary.split("万");
              String[ ] s2=s1[0].split("-");
                 double tmp1=Double.parseDouble(s2[0]);
                 double tmp2=Double.parseDouble(s2[1]);
                 average=(tmp1+tmp1)*10000/24;
                 }else if (jobSalary.contains("千")){
                       String[ ] s1=jobSalary.split("千");
                       String[ ] s2=s1[0].split("-");
                       double tmp1=Double.parseDouble(s2[0]);
                       double tmp2=Double.parseDouble(s2[1]);
                       average=(tmp1+tmp1)*1000/24;
                 }
             }else if (jobSalary.contains("月") && jobSalary.contains("-")){
                       choose1=true ;
                       if (jobSalary.contains("万")){
```

严禁网络传播本PDF，违者责任自负！

```
                String[ ] s1 = jobSalary. split("万");
                String[ ] s2 = s1[0]. split("-");
                double tmp1 = Double. parseDouble(s2[0]);
                double tmp2 = Double. parseDouble(s2[1]);
                average = (tmp1+tmp1) * 10000/2;
            } else if (jobSalary. contains("千")){
                String[ ] s1 = jobSalary. split("千");
                String[ ] s2 = s1[0]. split("-");
                double tmp1 = Double. parseDouble(s2[0]);
                double tmp2 = Double. parseDouble(s2[1]);
                average = (tmp1+tmp1) * 1000/2;
            }
        }
```

② 还需要对岗位名称进行归类，分出管理、销售、运维测试、开发与技术支持 5 类岗位，来对其分别进行基于大专与本科不同学历的薪资分析。这 5 类岗位，在本案例中是以字符串判断形式来区分的，具体的字符串关键词库如下：

```
//根据岗位名分类
    public static String[ ] GL = {"项目经理","总监","产品经理"};//管理类
    public static String[ ] XS = {"销售","客户","营销","商务"};//销售类
    public static String[ ] KF = {"开发","研发","研究","云计算","工程师","大数据","架构","python","java","分析","Web"};//开发类
    public static String[ ] YW = {"运维","实施","openstack","测试","运营","虚拟化","linux"};//运维测试类
    public static String[ ] ZC = {"技术支持","售前","售后","方案","解决","咨询"};//技术支持类
```

③ 通过类似的查询语句，结合上述的判断程序，得到的最终结果见表 4-6。

表 4-6 数据处理后的薪资信息表

| category | education | average | count |
|---|---|---|---|
| 销售 | 本科 | 9132 | 370 |
| 技术支持 | 本科 | 10677 | 579 |
| 管理 | 本科 | 14810 | 524 |
| 开发 | 本科 | 11629 | 1617 |
| 运维测试 | 本科 | 10488 | 545 |
| 销售 | 大专 | 7054 | 743 |

严禁网络传播本PDF，违者责任自负！

续表

| category | education | average | count |
|---|---|---|---|
| 开发 | 大专 | 6957 | 923 |
| 技术支持 | 大专 | 6703 | 278 |
| 运维测试 | 大专 | 6287 | 385 |
| 管理 | 大专 | 10225 | 187 |

## 项目实训

**【实训任务】**

利用 Java 远程连接 Hive 数据库，并进行数据处理。

**【实训目的】**

1. 熟悉 Java 连接 Hive 的基本步骤。
2. 学会 Java 对 Hive 的读、写数据操作。
3. 学会利用 HashMap 以及 HashSet 对数据进行处理。

**【实训内容】**

基于项目 3 中上传至 Hive 的数据表，利用 Java 进行远程连接，并定制过滤查询条件（查询岗位发布地在江苏，学历要求为大专的所有岗位的平均薪资）。

## 项目小结

数据预处理过程是大数据分析流程中一个非常关键的步骤，因为在后续数据分析的项目中，所需要的数据是规范化、结构化的。利用 MapReduce 可以很好地提高处理大数据的效率。当然，利用 MapReduce 在面对小数据量时，很可能是没有常规单机处理速度快。Hive 是基于 MapReduce 计算框架的，它提供了类似于 SQL 语句的查询重写方法，使得有数据库基础的开发者不需要去了解底层的 MapReduce 过程，就可以使用它来读写大量的数据。同样的，Hive 在处理小数据量时也没有优势。

非卖品，仅供非商业用途或交流学习使用

# 项目 5

# 岗位和技能数据的分析

PPT5：岗位和技能数据的分析

## 学习目标

【知识目标】

- 掌握 Mahout 算法库。
- 了解文本-向量转换的原理和 Kmeans + Canopy 算法机制。

【技能目标】

- 学会使用 Mahout 命令行实现文本聚类操作。
- 学会 Mahout 算法库中关于文本聚类的算法 Java API。
- 学会 Mahout 算法库中关于向量距离相似度算法的 Java API。

## 学习情境

在将文本数据进行了清洗和预处理后，米老师便可以进行大数据分析中最为重要的步骤了，即数据分析。首先需要做的是岗位技能分析。米老师希望从大量的岗位数据中挖掘得到以下信息：关于“云计算”的岗位究竟有哪些？这些岗位需要具备哪些技能？通过了解 Mahout 算法库，米老师决定采用 K-means 文本聚类的方法来进行数据挖掘。

在完成岗位和技能分析之后，米老师希望将得到的数据分析结果用于个性化的就业推荐与指导。计划采用向量的相似度算法进行匹配与推荐，借鉴基于用户推荐的算法机制。

非卖品，仅供非商业用途或交流学习使用

# 任务 5.1 使用 Mahout 命令进行岗位聚类分析

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习将语言数学化的相关技术知识（重点学习词向量的方法，它是用来将语言中的词进行数学化的一种方式）。
2. 学习 K-means 聚类算法的相关技术知识。
3. 学习 Canopy 聚类算法的相关技术知识。
4. 学习欧氏距离和向量空间余弦相似度算法的相关技术知识。
5. 使用 Mahout 命令，实现岗位聚类分析操作。

## 任务目标

1. 学会 K-means 聚类算法模型。
2. 学会 Canopy + K-means 聚类算法模型。

## 知识准备

笔 记

### 1. 岗位-技能点词向量

将爬取到的岗位信息匹配已建立的数据库中的技能点。利用数据清洗中的去重，将没有匹配到任何技能点的岗位信息视为无用文本数据，将其删除，最后保留岗位名称，以及匹配到的技能点，词与词之间用空格分开。

然而这样的文本向量是无法在 HDFS 上进行处理的，因此需要通过 TF-IDF 加权的方法将词文本进行转换，得到可处理的数据向量。

（1）TF-IDF 加权

TF-IDF 是一种用于信息检索与数据挖掘的常用加权技术，用以评估一字词对于一个文件集或一个语料库中的一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。除了 TF-IDF 以外，因特网上的搜索引擎还会使用基于链接分析的评级方法，以确定文件在搜寻结果中出现的顺序。

总之，一个词语在一篇文章中出现次数越多，同时在其他文档中出现次数越少，越能够代表该文章，这也就是 TF-IDF 的含义。

（2）词频（TF）

TF 是某一个给定的词语在该文件中出现的次数。这个数字通常会被归一化（一般是词频除以文章总词数），以防止它偏向长的文件（同一个词语在长文件里可能会比短文件有更高的词频，而不管该词语重要与否）。

非卖品，仅供非商业用途或交流学习使用

但需要注意，一些通用的词语对于主题并没有太大的作用，反倒是一些出现频率较少的词才能够表达文章的主题，所以单纯使用 TF 是不合适的。因此，权重的设计必须满足："一个词预测主题的能力越强，权重越大，反之，权重越小"。所有统计的文章中，一些词只是在其中很少几篇文章中出现，这样的词对文章的主题的作用很大，这些词的权重应该设计得较大。IDF 就是在完成这样的工作。

（3）逆文档频率（IDF）

IDF 的主要思想是，如果包含词条 t 的文档越少，IDF 越大，则说明词条具有很好的类别区分能力。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到。

算法实现步骤如下：

① 计算词频。

$$\text{词频(TF)} = \text{某个词在文章中的出现次数}$$

考虑到文章有长短之分，为了便于不同文章的比较，进行词频标准化。

$$\text{词频(TF)} = \frac{\text{某个词在文章中的出现次数}}{\text{文章的总词数}}$$

或者

$$\text{词频(TF)} = \frac{\text{某个词在文章中的出现次数}}{\text{该文出现次数最多的词的出现次数}}$$

② 计算逆文档频率。这时，需要一个语料库（corpus），用来模拟语言的使用环境。

$$\text{逆文档频率(IDF)} = \log\left(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数}+1}\right)$$

如果一个词越常见，那么分母就越大，逆文档频率就越小，其值越接近于 0。分母之所以要加 1，是为了避免分母为 0（即所有文档都不包含该词）。

③ 计算 TF-IDF。

$$\text{TF-IDF} = \text{词频(TF)} \times \text{逆文档频率(IDF)}$$

可以看到，TF-IDF 与一个词在文档中的出现次数成正比，与该词在整个语料库中的出现次数成反比。所以，自动提取关键词的算法就是计算出文档的每个词的 TF-IDF 值，然后按降序排列，取排在最前面的几个词。

### 2. K-means 聚类算法

在将岗位-技能点的向量转换为 HDFS 可处理的数据后，便可以进行基于文本词频与 TF-IDF 的 K-means 聚类。

K-means 算法是最为经典的基于划分的聚类方法。K-means 算法的基本思想是以空间中 K 个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

K-means 算法接受参数 K，然后将事先输入的 N 个数据对象划分为 K 个聚类以便使得所获得的聚类满足"同一聚类中的对象相似度较高，而不同聚类中的对象相似度

非卖品，仅供非商业用途或交流学习使用

较小”。聚类相似度是利用各聚类中对象的均值所获得一个“中心对象（引力中心）”来进行计算的。

算法实现步骤如下：

① 从 N 个文档随机选取 K 个文档作为质心。

输入：k，data［n］；

选择 k 个初始中心点，如 c［0］=data［0］，…c［k-1］=data［k-1］；

② 对剩余的每个文档测量其到每个质心的距离，并把它归到最近的质心的类。

对于 data［0］…data［n］，分别与 c［0］…c［k-1］比较，假定与 c［i］差值最少，就标记为 i。

③ 重新计算已经得到的各个类的质心。

对于所有标记为 i 点，重新计算 c［i］=｛所有标记为 i 的 data［j］之和｝/标记为 i 的个数。

④ 迭代步骤②~③直至新的质心与原质心相等或小于指定阈值，算法结束。

重复步骤②、③，直到所有 c［i］值的变化小于给定阈值。

笔记

上述算法实现中需要注意的问题如下：

（1）K 值的确定

K-means 算法首先选择 K 个初始质心，其中 K 是用户指定的参数，即所期望簇的个数。这样做的前提是已经知道数据集中包含多少个簇，但在很多情况下，并不知道数据的分布情况，实际上聚类就是发现数据分布的一种手段，这就陷入了鸡和蛋的矛盾。如何有效地确定 K 值，大致可以提供以下几种方法：

1）与层次聚类结合。经常会产生较好的聚类结果的一个有趣策略是，首先采用层次聚类算法决定结果粗略的数目，并找到一个初始聚类，然后用迭代重定位来改进该聚类。

2）稳定性方法。其对一个数据集进行两次重采样产生两个数据子集，再用相同的聚类算法对两个数据子集进行聚类，产生两个具有 K 个聚类的聚类结果，计算两个聚类结果的相似度的分布情况。两个聚类结果具有高的相似度说明 K 个聚类反映了稳定的聚类结构，其相似度可以用来估计聚类个数。采用此方法试探多个 K，找到合适的 K 值。

3）系统演化方法。其将一个数据集视为伪热力学系统，当数据集被划分为 K 个聚类时称系统处于状态 K。系统由初始状态 K=1 出发，经过分裂和合并过程，系统将演化到它的稳定平衡状态 Ki，其所对应的聚类结构决定了最优类数 Ki。系统演化方法能提供关于所有聚类之间的相对边界距离或可分程度，它适用于明显分离的聚类结构和轻微重叠的聚类结构。

4）使用 Canopy 算法进行初始划分。基于 Canopy 算法的聚类算法将聚类过程分为以下两个阶段：

① 聚类最耗费计算的地方是计算对象相似性的时候，Canopy 算法在第一阶段选择简单、计算代价较低的方法计算对象相似性，将相似的对象放在一个子集中，该子集被叫作 Canopy，通过一系列计算得到若干 Canopy，Canopy 之间可以是重叠的，但

非卖品，仅供非商业用途或交流学习使用

不会存在某个对象不属于任何 Canopy 的情况，可以把这一阶段看作数据预处理。

② 在各个 Canopy 内使用传统的聚类方法（如 K-means），不属于同一 Canopy 的对象之间不进行相似性计算。

该方法起码有两点优势：第一，Canopy 不要太大且 Canopy 之间重叠的不要太多情况会大大减少后续需要计算相似性的对象的个数；第二，类似于 K-means 这样的聚类方法是需要人为指出 K 的值的，通过第一阶段得到的 Canopy 个数完全可以作为这个 K 值，一定程度上减少了选择 K 的盲目性。

（2）初始质心的选取

选择适当的初始质心是基本 K-means 算法的关键步骤。常见的方法是随机地选取初始质心，但是这样簇的质量常常很差。处理选取初始质心问题的常用技术如下：

1）多次运行，每次使用一组不同的随机初始质心，然后选取具有最小 SSE（误差的平方和）的簇集。这种策略简单，但效果可能不好，其取决于数据集和寻找的簇的个数。

2）取一个样本，并使用层次聚类技术对它聚类。从层次聚类中提取 K 个簇，并用这些簇的质心作为初始质心。该方法通常很有效，但仅对下列情况有效：样本相对较小，如数百到数千（层次聚类开销较大）；K 相对于样本大小较小。

3）随机地选择第一个点，或取所有点的质心作为第一个点。然后，对于每个后继初始质心，选择离已经选取过的初始质心最远的点。使用这种方法，确保了选择的初始质心不仅是随机的，而且是散开的。但是，这种方法可能选中离群点。此外，求离当前初始质心集最远点的开销也非常大。为了克服这个问题，通常该方法用于点样本。由于离群点很少（多了就不是离群点了），它们多半不会在随机样本中出现，计算量大幅减少。

4）Canopy 算法。

（3）距离的度量

常用的距离度量方法包括欧氏距离和向量空间余弦相似度。两者都是评定个体间差异大小的。欧氏距离度量会受指标不同单位刻度的影响，其一般需要先进行标准化，同时距离越大，个体间差异越大。向量空间余弦相似度的度量不会受指标刻度的影响，余弦值落于区间[-1,1]，值越大，差异越小。

针对具体应用，在不同情况下使用欧氏距离或余弦相似度有不同要求。从几何意义上来说，n 维向量空间的一条线段作为底边和原点组成的三角形，其顶角大小是不确定的。也就是说对于两条空间向量，即使两点距离一定，它们的夹角余弦值也可以随意变化。感性地认识，当两用户评分趋势一致时，但是评分值差距很大，余弦相似度倾向给出更优解。举个极端的例子，两用户只对两件商品评分，向量分别为(3，3)和（5，5），这两位用户的认知其实是一样的，但是欧式距离给出的解显然没有余弦值合理。

（4）质心的计算

对于距离度量不管是采用欧式距离还是采用余弦相似度，簇的质心都是其均值，即向量各维取平均即可。对于距离度量采用其他方式时，读者可根据情况自主开展相

非卖品，仅供非商业用途或交流学习使用

关研究。

（5）算法停止条件

一般是目标函数达到最优或者达到最大的迭代次数即可终止。对于不同的距离度量，目标函数往往不同。

当采用欧式距离时，目标函数一般为最小化对象到其簇质心的距离的平方和：

$$\min \sum_{i=1}^{K} \sum_{x \in C_i} \mathrm{dist}(c_i, x)^2$$

当采用余弦相似度时，目标函数一般为最大化对象到其簇质心的余弦相似度和：

$$\max \sum_{i=1}^{K} \sum_{x \in C_i} \mathrm{cosine}(c_i, x)$$

（6）空聚类的处理

如果所有的点在指派步骤都未分配到某个簇，就会得到空簇。如果这种情况发生，则需要某种策略来选择一个替补质心，否则，其平方误差将会偏大。一种方法是选择一个距离当前任何质心最远的点，这将消除当前对总平方误差影响最大的点。另一种方法是从具有最大 SSE 的簇中选择一个替补的质心，这将分裂簇并降低聚类的总 SSE。如果有多个空簇，则该过程重复多次。另外，通过编程实现时，要注意空簇可能导致的程序 bug。

K-menas 算法试图找到使平均误差准则函数最小的簇。当潜在的簇形状是凸面的，簇与簇之间区别较明显，且簇大小相近时，其聚类结果较理想。该算法时间复杂度为 O（tKmn），与样本数量线性相关，因此，对于处理大数据集合，该算法非常高效，且伸缩性较好。但该算法除了要事先确定簇数 K 和对初始聚类中心敏感外，经常以局部最优结束，同时对“噪声”和孤立点敏感，且该方法不适于发现非凸面形状的簇或大小差别很大的簇。

### 3. Canopy 聚类算法

之前具体讨论了 K-means 的一些注意问题，其中与算法是否可以有效实现关系最为密切的两个问题，便是 K 的不确定性与初始化聚类中心的选取。这是由于事先并不知道招聘网站上具体有多少个与云计算大数据相关的岗位，所以无法事先就判定应该聚成多少类。因此需要改进原始的 K-means 算法来实现，这里采用 Canopy 算法来进行初始化聚类中心的选取，从而可以不设定 K 值。

在生活中，可以使用聚类解决很多问题。传统的聚类算法对于一般的应用问题（基本都是小数据量）都是可以解决的，但是当数据变得很大的时候，就有点力不从心了，这里的数据变得很大指的是：

① 数据的条目很多，整个数据集包含的样本数据向量很多。

② 针对①中的每个样本数据向量其维度很大，即包含多个属性。

③ 要聚类的中心向量很多。当所要应用聚类算法的数据是上述情况时，传统的聚类方法应用起来就会相当棘手，这时就要采取另外的途径，即改进的聚类算法。任务 5.1 中介绍的 Canopy 算法就是聚类算法发展到一定阶段，Andrew McCallum、Kamal

非卖品，仅供非商业用途或交流学习使用

笔记

Nigam、Lyle H. Ungar 根据前人经验加上自己的想法提出来的一种改进算法。

Canopy 算法的主要思想是把聚类分为两个阶段：第 1 阶段，通过使用一个简单、快捷的距离计算方法把数据分为可重叠的子集，称为 Canopy；第 2 阶段，通过使用一个精准、严密的距离计算方法来计算出现在阶段一中同一个 Canopy 的所有数据向量的距离。这种方式和之前的聚类方式不同的地方在于使用了两种距离计算方式，同时因为只计算了重叠部分的数据向量，所以达到了减少计算量的目的。

具体来说：

第 1 阶段，使用一个简单距离计算方法来产生具有一定数量的可重叠的子集。Canopy 就是一个样本数据集的子集，子集中的样本数据是通过一个粗糙的距离计算方法来计算样本数据向量和 Canopy 的中心向量的距离，设定一个距离阈值，当计算的距离小于该阈值时，就把样本数据向量归为此 Canopy。这里要说明的是，每个样本数据向量有可能存在于多个 Canopy 里面，但是每个样本数据向量至少要包含于一个 Canopy 中。Canopy 的创建基于不在同一个 Canopy 中的样本数据向量彼此不相似，不能被分为同一个类的观点去考虑。由于距离计算方式是粗糙的，因此不能够保证性能（计算精确度）。但是通过允许存在可叠加的 Canopy 和设定一个较大的距离阈值，在某些情况下可以保证该算法的性能。

那么针对一个具体的 Canopy 应该如何创建呢？以下是创建一个普通的 Canopy 算法的简单步骤，如图 5-1 所示。

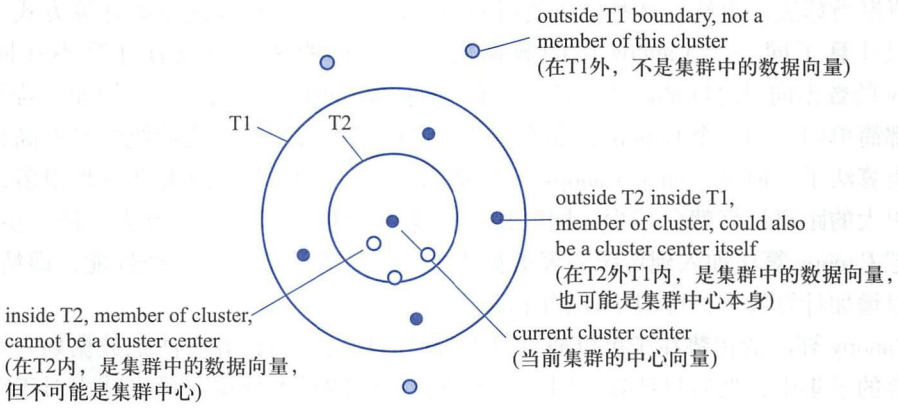



图 5-1　Canopy 聚类的概念图

① 原始数据集合 List 按照一定的规则进行排序（该规则是任意的，但一旦确定就不再更改），初始距离阈值为 T1、T2，且 T1 > T2（T1、T2 的设定可以根据用户的需要，或者使用交叉验证获得）。

② 在 List 中随机挑选一个数据向量 A，使用一个粗糙距离计算方式计算 A 与 List 中其他样本数据向量之间的距离 d。

③ 根据②中的距离 d，把 d 小于 T1 的样本数据向量划到一个 Canopy 中，同时把 d 小于 T2 的样本数据向量从候选中心向量名单（这里可以理解为 List）中移除。

④ 重复步骤②和③，直到候选中心向量名单为空，即 List 为空，算法结束。

非卖品，仅供非商业用途或交流学习使用

如图 5-2 所示为创建 Canopy 算法的流程图。

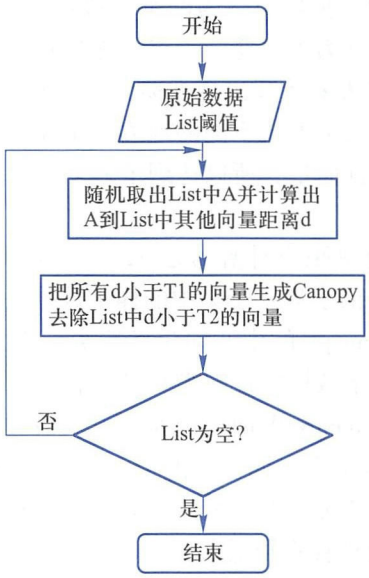



图 5-2 Canopy 算法流程图

第 2 阶段，可以在第 1 阶段的基础上应用传统聚类算法，如贪婪凝聚聚类算法、K 均值聚类算法，当然，这些算法使用的距离计算方式是精准的距离计算方式。但是因为只计算了同一个 Canopy 中的数据向量之间的距离，而没有计算不在同一个 Canopy 的数据向量之间的距离，所以假设它们之间的距离为无穷大。例如，若所有的数据都简单归入同一个 Canopy，那么第 2 阶段的聚类就会退化成传统的具有高计算量的聚类算法了。但是，如果 Canopy 不是那么大，且它们之间的重叠不是很多，那么代价很大的距离计算就会减少，同时用于分类的大量计算也可以省去。进一步来说，如果把 Canopy 算法加入到传统的聚类算法中，那么算法既可以保证性能，即精确度，又可以增加计算效率，即减少计算时间。

Canopy 算法的优势在于可以通过第 1 阶段的粗糙距离计算方法把数据划入不同的可重叠的子集中，然后只计算在同一个重叠子集中的样本数据向量来减少对于需要距离计算的样本数量。

微课：5-1 使用 Mahout 命令进行岗位聚类分析

## 任务实施

由于招聘网站上的数据非常庞大，所抓取到的文本信息甚至需要以 TB 为单位来存储，所以在进行大数据量分析代码实现之前，首先需要先对小数据集进行测试，以验证算法的可行性。

现从 51job 上收集了 30 条岗位信息，其中 10 条是大数据开发工程师，10 条是云计算开发工程师，10 条是 Web 开发工程师，每一个岗位信息都用一个 txt 文件保存，编码格式为 UTF-8（与 Linux 默认编码格式一致）。

由于已知这 30 个测试数据集应该聚成 3 类，所以可以直接使用 K-means 算法来

非卖品，仅供非商业用途或交流学习使用

实现，也可以进行 Canopy 结合 K-means 算法来实现，在本任务将采用这两种方法进行实现。

代码 5.1：BigDataStudyAnalysis——项目5

（1）以 K-means 实现

① 数据准备。版本说明：使用的 Mahout 版本为 apache-mahout-distribution-0.12.2，使用的 Hadoop 版本为 hadoop-2.4.1，将文本文件转换成 Sequence File。

将 30 个 txt 文件压缩后上传至远程虚拟机，解压至一个本地目录，如/tmp/cetc55/txt。

使用 Mahout 命令将文本文件转换成 SequenceFile，同时会将这么多小文本文件合并成一个 Sequence。

```
[root@master ~]# cd $MAHOUT_HOME/bin
[root@master bin]# export MAHOUT_LOCAL=true
[root@master bin]# mahout seqdirectory -i file:///tmp/cetc55/txt/ -o file:///tmp/cetc55/txt-seq/ -c UTF-8 -chunk 64 -xm sequential
```

参数说明：

MAHOUT_LOCAL：指定 Mahout 是否以本地模式运行；如果该变量值为空，则表示在 Hadoop 上运行。

seqdirectory：Mahout 自带的将文本文件转换成 SequenceFile 的命令。

-i：输入的文本文件目录。

-o：输出的 SequenceFile 目录。

-xm sequential：表示在本地执行，而不是用 MapReduce 执行。

-c UTF8：使用 UTF8 编码格式。

-chunk 64：64 MB 一个 chunk，也就是文件块 Block，应该和 HDFS 的块大小一致或成倍数关系。

执行成功后，在本地的/tmp/cetc55/txt-seq 目录下会生成 chunk-0 文件，该文件格式为 SequenceFile，可以使用“hadoop fs -text”命令查看文件内容。

```
[root@master bin]# hadoop fs -text file:///tmp/cetc55/txt-seq/chunk-0 | more
/23.txt    Web 开发工程师 mvc html js jquery css
/7.txt    大数据开发工程师 oracle mysql java shell python php ruby hadoop spark mapreduce
hive sparksql kylin
/14.txt    云计算研发工程师 虚拟化 sds sdn linux windows openstack swift cinder python java shell
/18.txt    云计算研发工程师 虚拟化 vmware kvm openstack
/12.txt    云计算研发工程师 openstack ubuntu centos linux kvm python ruby java c c++ shell ceph
/3.txt    大数据开发工程师 数据分析 mapreduce 数据清洗 storm hadoop hive hbase spark nosql java mongodb redis cassandra linux
```

将文本文件转换成 SequenceFile，可以自己写程序来完成，Mahout 也提供了相应 API。将转换好的 SequenceFile 上传至 HDFS。

```
[root@ master bin]# hadoop fs -mkdir -p /tmp/cetc55/txt-seq/
[root@ master bin]# hadoop fs -put /tmp/cetc55/txt-seq/chunk-0 /tmp/cetc55/txt-seq/
```

将 SequenceFile 上传至 HDFS 的/tmp/cetc55/txt-seq/目录。

② 解析 SequenceFile，转换成向量表示。Mahout 聚类算法使用向量空间（Vectors）作为数据。接下来在 Hadoop 上，使用之前生成的 SequenceFile，转换成向量表示。

```
[root@ master bin]# cd $MAHOUT_HOME/bin
[root@ master bin]# export MAHOUT_LOCAL=
[root@ master bin]# export HADOOP_CLASSPATH=/opt/hadoop_resources/apache-mahout-distribution-0.12.2/lib/lucene-analyzers-common-4.6.1.jar:/opt/hadoop_resources/apache-mahout-distribution-0.12.2/lib/lucene-core-4.6.1.jar:$HADOOP_CLASSPATH
[root@ master bin]# mahout seq2sparse -i /tmp/cetc55/txt-seq/ -o /tmp/cetc55/txt-sparse -ow --weight tfidf --maxDFPercent 99 --namedVector -a org.apache.lucene.analysis.core.WhitespaceAnalyzer
```

参数说明：

MAHOUT_LOCAL：由于要在 Hadoop 上完成转换，因此，设该变量值为空。

HADOOP_CLASSPATH：加入需要的 jar 包到 Hadoop 的 classpath，这两个 jar 包主要是 Mahout 用来分词的。

seq2sparse：Mahout 自带的解析向量的命令。

-i：输入的 SequenceFile 目录。

-o：输出目录。

-ow：当输出目录存在时，覆盖。

-weight tfidf：权重公式，可选的有 tfidf 和 tf。

-maxDFPercent：过滤高频词，当 DF 值大于 90%时候，将会被过滤掉。

-namedVector：生成的向量同时输出附加信息。

-a：指定分词器；由于 Mahout 本身对中文支持不好，前面已经将中文做好了分词，以空格分隔，这里指定的 WhitespaceAnalyzer 是以空格为分隔符的分词器。

其他可能有用的选项如下：

-minDF：最小 DF 阈值。

-minSupport：最小的支持度阈值，默认为 2。

-maxNGramSize：是否创建 ngram，默认为 1。建议一般设定为 2。

-minLLR：The minimum Log Likelihood Ratio。默认为 1.0。当设定了-ng > 1 后，建议设置为较大的值，只过滤有意义的 N-Gram。

-logNormalize：是否对输出向量做 Log 变换。

其他可输入命令“mahout seq2sparse-help”来查看。

成功运行的过程中部分 INFO 文档展示如下。

```
17/03/07 12:41:49 INFO RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/03/07 12:41:50 INFO FileInputFormat: Total input paths to process : 1
17/03/07 12:41:50 INFO JobSubmitter: number of splits:1
17/03/07 12:41:50 INFO JobSubmitter: Submitting tokens for job: job_1487107260988_0079
17/03/07 12:41:50 INFO YarnClientImpl: Submitted application application_1487107260988_0079
17/03/07 12:41:50 INFO Job: The url to track the job: http://master:8088/proxy/application_1487107260988_0079/
17/03/07 12:41:50 INFO Job: Running job: job_1487107260988_0079
17/03/07 12:42:08 INFO Job: Job job_1487107260988_0079 running in uber mode : false
17/03/07 12:42:08 INFO Job:  map 0% reduce 0%
17/03/07 12:42:17 INFO Job:  map 100% reduce 0%
17/03/07 12:42:27 INFO Job:  map 100% reduce 100%
17/03/07 12:42:28 INFO Job: Job job_1487107260988_0079 completed successfully
17/03/07 12:42:28 INFO Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=3043
        FILE: Number of bytes written=193595
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=3462
        HDFS: Number of bytes written=3326
        HDFS: Number of read operations=7
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Data-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=7203
```

```
        Total time spent by all reduces in occupied slots (ms)=8055
        Total time spent by all map tasks (ms)=7203
        Total time spent by all reduce tasks (ms)=8055
        Total vcore-seconds taken by all map tasks=7203
        Total vcore-seconds taken by all reduce tasks=8055
        Total megabyte-seconds taken by all map tasks=7375872
        Total megabyte-seconds taken by all reduce tasks=8248320
    Map-Reduce Framework
        Map input records=30
        Map output records=30
        Map output bytes=2976
        Map output materialized bytes=3043
        Input split bytes=136
        Combine input records=0
        Combine output records=0
        Reduce input groups=30
        Reduce shuffle bytes=3043
        Reduce input records=30
        Reduce output records=30
        Spilled Records=60
        Shuffled Maps=1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=166
        CPU time spent (ms)=2170
        Physical memory (bytes) snapshot=323907584
        Virtual memory (bytes) snapshot=1689812992
        Total committed heap usage (bytes)=164302848
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=3326
```

```
        File Output Format Counters
                Bytes Written=3326
    17/03/07 12:42:29 INFO HadoopUtil: Deleting /tmp/cetc55/txt-sparse/partial-
vectors-0
    17/03/07 12:42:29 INFO MahoutDriver: Program took 360363 ms (Minutes:
6.00605)
```

Seq2sparse 运行过程中会进行若干次的 MapReduce，生成若干个文件，这是最后一次 MapReduce 运行成功的 INFO 文档信息。

在 HDFS 上查看解析后的结果如下：

```
[root@master bin]# hadoop fs -ls /tmp/cetc55/txt-sparse
Found 7 items
drwxr-xr-x   - root supergroup      0 2017-03-07 12:39 /tmp/cetc55/txt-sparse/df
-count
-rw-r--r--   1 root supergroup    956 2017-03-07 12:37 /tmp/cetc55/txt-sparse/
dictionary.file-0
-rw-r--r--   1 root supergroup    993 2017-03-07 12:39 /tmp/cetc55/txt-sparse/
frequency.file-0
drwxr-xr-x   - root supergroup      0 2017-03-07 12:41 /tmp/cetc55/txt-sparse/tf
-vectors
drwxr-xr-x   - root supergroup      0 2017-03-07 12:42 /tmp/cetc55/txt-sparse/
tfidf-vectors
drwxr-xr-x   - root supergroup      0 2017-03-07 12:36 /tmp/cetc55/txt-sparse/
tokenized-documents
drwxr-xr-x   - root supergroup      0 2017-03-07 12:37 /tmp/cetc55/txt-
sparse/wordcount
```

各个文件的用途如下：

file-0：文件。词文本→词 id 的映射。词转化为 id 是常见做法。

```
[root@master bin]# hadoop fs -text /tmp/cetc55/txt-sparse/dictionary.file-0 | more
ajax                    0
bootstrap               1
c   2
c++                     3
ceph                    4
css                     5
css3                    6
div                     7
```

```
docker          8
elasticsearch       9
hadoop          10
```

df-count：目录。词 id→文档频率（DF）。

file：词 id →文档集词频（CF）。

wordcount：目录。词文本→ 文档集词频（CF），是各种过滤处理之前的信息。

```
[root@ master bin]# hadoop fs -text /tmp/cetc55/txt-sparse/wordcount/ * | more
ajax        7
bootstrap   5
c           3
c++         5
ceph        2
css         8
css3        4
div         2
docker      2
elasticsearch       2
hadoop      10
```

tf-vectors、tfidf-vectors：目录。词向量，每篇文档一行，格式为｛词 id：特征值｝，其中特征值为 tf 或 tfidf。采用了内置类型 VectorWritable，需要使用命令“mahout vectordump -i <path>”查看。

```
[root@ master bin]# mahout vectordump -i /tmp/cetc55/txt-sparse/tf-vectors/
| more
{33:1.0,29:1.0,34:1.0,44:1.0,9:1.0,20:1.0,10:1.0}
{34:1.0,25:1.0,15:1.0,12:1.0,44:1.0,21:2.0,11:1.0,10:1.0}
{33:1.0,3:1.0,29:1.0,2:1.0,26:1.0,43:1.0,20:2.0}
{33:1.0,29:1.0,2:1.0,4:1.0,15:1.0,31:1.0,3:1.0,26:1.0,19:1.0,43:1.0,
20:1.0}
{4:1.0,15:1.0,8:1.0,32:1.0,38:1.0,26:1.0,19:1.0,43:1.0,20:1.0}
{33:1.0,29:1.0,26:1.0,15:1.0,41:1.0,43:1.0,20:1.0,32:1.0}
{32:1.0,29:1.0,24:1.0,26:1.0,19:1.0,41:1.0,43:1.0,20:1.0}
{29:1.0,39:1.0,37:1.0,26:1.0,19:1.0,41:1.0,43:1.0,20:1.0}
{3:1.0,29:1.0,38:1.0,26:1.0,15:1.0,43:1.0,20:1.0}
{37:1.0,26:1.0,19:1.0,41:1.0,43:1.0}
{33:1.0,29:1.0,39:1.0,37:1.0,26:1.0,19:1.0,41:1.0,43:1.0,20:1.0}
```

```
[root@master bin]# mahout vectordump -i /tmp/mahout/txt-sparse/tfidf-vectors/
| more
{33:2.321755886077881,29:1.8362480401992798,34:2.0986123085021973,44:
2.0033020973205566,9:3.3025851249694824,20:1.6931471824645996,10:2.00330
20973205566}
{34:2.0986123085021973,25:2.791759490966797,15:1.6931471824645996,12:
2.321755886077881,44:2.0033020973205566,21:3.6903023719787598,11:2.203972
816467285,10:2.0033020973205566}
{33:2.321755886077881,3:2.609437942504883,29:1.8362480401992798,2:
3.0149030685424805,26:2.0033020973205566,43:2.0033020973205566,20:2.39447
16453552246}
{33:2.321755886077881,29:1.8362480401992798,2:3.0149030685424805,4:
3.3025851249694824,15:1.6931471824645996,31:3.0149030685424805,3:2.6094379
42504883,26:2.0033020973205566,19:2.455287218093872,43:2.0033020973205566,
20:1.6931471824645996}
{4:3.3025851249694824,15:1.6931471824645996,8:3.3025851249694824,32:
3.0149030685424805,38:2.0033020973205566,26:2.0033020973205566,19:2.455287
218093872,43:2.0033020973205566,20:1.6931471824645996}
```

tokenized-documents：分词后的文档。

③ 运行 K-means。这里使用 tf-vectors 作为输入文件，即根据关键词出现的频率来进行聚类。

```
[root@master bin]# cd $MAHOUT_HOME/bin
[root@master bin]# export MAHOUT_LOCAL=
[root@master bin]# mahout kmeans -i /tmp/cetc55/txt-sparse/tf-vectors -c
/tmp/cetc55/txt-kmeans-clusters -o /tmp/cetc55/txt-kmeans -k 3 -dm
org.apache.mahout.common.distance.CosineDistanceMeasure -x 100 -ow --clus-
tering --convergenceDelta 0.3
```

参数说明：

-i：输入为上面产出的 tf 向量。

-o：在这里输出每一轮迭代的结果。

-k：几个簇，即最终聚为几个类别。

-c：若不设定 K，则用该目录中的点作为聚类中心点。否则，随机选择 K 个点作为中心点。

-dm：距离公式，文本类型推荐用 Cosine 距离。

-x：最大迭代次数。

-clustering：用 MapReduce 模式运行。

-convergenceDelta：迭代收敛阈值，默认 0.5，对于 Cosine 来说略大，因此这里采

用 0.3。

输出 1：初始随机选择的中心点。

```
[root@master bin]# hadoop fs -ls /tmp/cetc55/txt-kmeans-clusters
Found 1 items
-rw-r--r-- 1 root supergroup 620 2017-03-07 13:26 /tmp/cetc55/txt-kmeans-clusters/part-randomSeed
```

输出 2：聚类过程、结果。

```
[root@master bin]# hadoop fs -ls /tmp/cetc55/txt-kmeans
Found 5 items
-rw-r--r-- 1 root supergroup 194 2017-03-07 14:07 /tmp/cetc55/txt-kmeans/_policy
drwxr-xr-x - root supergroup 0 2017-03-07 14:08 /tmp/cetc55/txt-kmeans/clusteredPoints
drwxr-xr-x -root supergroup 0 2017-03-07 14:06 /tmp/cetc55/txt-kmeans/clusters-0
drwxr-xr-x -root supergroup 0 2017-03-07 14:07 /tmp/cetc55/txt-kmeans/clusters-1
drwxr-xr-x -root supergroup 0 2017-03-07 14:07 /tmp/cetc55/txt-kmeans/clusters-2-final
```

其中，clusters-n(-final)为每次迭代后，簇的 3 个中心点的信息。

④ 查看聚类结果。由于 Mahout 的 clusterdump 命令只能在本地运行，因此需要将聚类结果从 HDFS 下载到本地。

```
[root@master bin]# hadoop fs -get /tmp/cetc55/txt-kmeans/ /tmp/cetc55
[root@master bin]# hadoop fs -get /tmp/cetc55/txt-sparse/ /tmp/cetc55
[root@master bin]# cd $MAHOUT_HOME/bin
[root@master bin]# export MAHOUT_LOCAL=true
[root@master bin]# mahout clusterdump -i /tmp/cetc55/txt-kmeans/clusters-2-final -d /tmp/cetc55/txt-sparse/dictionary.file-0 -dt sequencefile -o /tmp/cetc55/txt-kmeans-cluster-dump -n 15
```

参数说明：

-i：输入最终迭代生成的簇结果。

-d：使用词→词 id 映射，使得输出结果中可以直接显示每个簇，权重最高的词文本，而不是词 id。

-dt：上面映射类型。

-o：最终产出目录。

-n：每个簇，只输出 15 个权重最高的词。

将文件下载到 PC 端本地并用 notepad+查看 dump 出来的结果。

```
Top Terms:
        linux                   =>              0.9
        python                  =>              0.8
        shell                   =>              0.7
        openstack               =>              0.7
        云计算研发工程师                =>              0.7
        kvm                     =>              0.5
        虚拟化                     =>              0.5
        vmware                  =>              0.3
        java                    =>              0.3
        hadoop                  =>              0.3
        大数据开发工程师                =>              0.3
        hive                    =>              0.2
        mysql                   =>              0.2
        mapreduce               =>              0.2
        xen                     =>              0.2
Top Terms:
        Web 开发工程师                 =>              1.0
        css                     =>              0.8
        ajax                    =>              0.7
        javascript              =>              0.7
        Web                     =>              0.7
        jquery                  =>              0.6
        js                      =>              0.6
        html5                   =>              0.6
        html                    =>              0.5
        bootstrap               =>              0.5
        css3                    =>              0.4
        mysql                   =>              0.3
        mvc                     =>              0.3
        php                     =>              0.2
        sql                     =>              0.2
Top Terms:
        java                    =>              1.0
        spark                   =>              0.8
        hbase                   =>              0.7
```

```
        hadoop                  =>      0.7
        大数据开发工程师          =>      0.7
        hive                    =>      0.5
        linux                   =>      0.5
        python                  =>      0.4
        nosql                   =>      0.4
        mysql                   =>      0.4
        mapreduce               =>      0.4
        storm                   =>      0.4
        redis                   =>      0.3
        openstack               =>      0.3
        Web                     =>      0.3
```

可以看出，一共有 3 个 Top Terms，即聚类成了 3 类。文本的形式是词→词频，在每个聚类中，出现次数越高的词，排名越靠前，有可能是技能点，也有可能是岗位名称，但一般来说，岗位名称的排名都会比较靠前。

先观察第 2 个聚类，排名最高的词汇是“Web 开发工程师”，词频为 1.0，即该聚类每一个文档都出现了，说明在 K-means 聚类下，该聚类收敛的情况非常好，其中只有 Web 工程师的信息，相应也可以得到最相关的一些技能点，如 CSS、Ajax、Javascript、Web、Jquery、JS、HTML5 等。

再观察第 3 个聚类，排名在第 5 位的是“大数据开发工程师”，说明该聚类是关于大数据开发工程师的，但是它的岗位名称词频并不是为 1.0，说明该聚类中还有其他岗位的信息，而对比第 1 个聚类，很容易发现第 3 个聚类中可能有一些云计算开发工程师的信息，而事实上这两个岗位的技能点的确有一部分是重合的，如 Linux、Java 等，说明聚类结果看似不是很理想，但其实是完全符合实际情况的。

但是在运行 K-means 时使用的是基于词频进行的聚类，可以认为当已经提取了关键词后，每一个词都是重要的，不会存在高频无用词，看似不是很理想的测试结果需要重新反思这个选择，假如使用基于词频-逆向文件频率进行测试，结果将会怎样呢？

运行命令几乎一样，不再赘述，直接给出最后输出的聚类结果如下。

```
Top Terms:
        openstack                       =>  2.0033020973205566
        云计算研发工程师                  =>  2.0033020973205566
        python                      =>  1.5023847601630471
        linux                       =>  1.449059009552002
        虚拟化                        =>  1.4233297868208452
        kvm                         =>  1.3392475735057483
        sdn                         =>  1.096328388560902
        shell                       =>  1.055343584580855
```

```
        java                    =>  0.9235348267988726
        vmware                  =>  0.8222462914206765
        c++                     =>  0.7116648934104226
        ceph                    =>  0.6004700227217241
        docker                  =>  0.6004700227217241
        xen                     =>  0.6004700227217241
        c                       =>  0.548164194280451
Top Terms:
        大数据开发工程师                    =>  2.0033020973205566
        hadoop                  =>  1.8211837248368696
        hbase                   =>  1.803250486200506
        spark                   =>  1.796071312644265
        hive                    =>  1.688549735329368
        mapreduce               =>  1.5215901895002886
        storm                   =>  1.4233297868208452
        java                    =>  1.2313797690651633
        nosql                   =>  1.0151852694424717
        linux                   =>  0.9235348267988726
        shell                   =>  0.844274867664684
        mysql                   =>  0.7631317485462535
        redis                   =>  0.7613889520818536
        python                  =>  0.6677265600724653
        elasticsearch           =>  0.6004700227217241
Top Terms:
        Web 开发工程师                    =>  2.0033020973205566
        css                     =>  1.803250486200506
        ajax                    =>  1.688549735329368
        javascript              =>  1.688549735329368
        Web                     =>  1.4569469798694958
        bootstrap               =>  1.4233297868208452
        jquery                  =>  1.3392475735057483
        html5                   =>  1.3392475735057483
        js                      =>  1.284368558363481
        html                    =>  1.1861081556840376
        css3                    =>  1.0151852694424717
        xml                     =>  0.9007050340825861
        php                     =>  0.8222462914206765
```

版权所有，严禁传播，违者自负法律责任！



```
mvc                                    =>  0.8222462914206765
sql                                    =>  0.8222462914206765
```

结果非常理想，每个聚类都非常纯粹，几乎没有两个岗位的信息重合到一个聚类的现象，说明此案例下，TF-IDF 相比 TF 加权方法，更加能够提取出文本的特征值，从而使得聚类效果达到理想的结果。

这证明了基于词频-逆向文件频率（TF-IDF）的文本聚类方法是非常有效的。

（2）以 Canopy+K-means 实现

以上测试了 K-means 算法分别基于 TF 与 TF-IDF 加权方法来实现，两者实验效果都很不错，TF-IDF 更为理想。但其前提是最后的聚类数已知，在 K-means 算法中已经假定了 K 值为 3。然而在现实情况，数以万计的岗位信息中，究竟能够最后聚成多少个岗位，并不能准确地事先得知，因此 K 值是无法预知的，K 值的确定也是 K-means 方法最为复杂的问题之一。

Mahout 的 K-means 算法提供了另一种选择，即如果提供了初始化聚类中心，那么便不需要设定 K 值。那么问题又来了，初始化聚类中心如何得到呢？Canopy 算法便是一个有效的方法，并且 Mahout 有相应封装好的 jar 包，可以直接从命令行进行操作，实现起来非常方便。

① 准备工作。Canopy 算法只需要在 K-means 之前插入即可，也就是说，在 seq2sparse 中得到的结果（txt-sparse）可以直接使用，可参考 Canopy 算法之前的准备工作。

② 运行 Canopy。在这里 TF-IDF 方法更为有效，此处仅使用 tfidf-vectors 进行测试。

运行命令：

```
[root@master ~]# cd $MAHOUT_HOME/bin
[root@master bin]# export MAHOUT_LOCAL=
[root@master bin]# mahout canopy -i /tmp/cetc55/txt-sparse/tfidf-vectors -o /tmp/cetc55/canopy-centroids -dm org.apache.mahout.common.distance.CosineDistanceMeasure -t1 0.7 -t2 0.4 -ow
```

参数说明：

-i：输入路径。

-o：输出目录。

--distanceMeasure(-dm)：距离度量类的权限命名，如 org.apache.mahout.common.distance.CosineDistanceMeasure。

--t1（-t1）：t1 值(t1>t2)。

--t2（-t2）：t2 值。

--t3（-t3）：t3 值，默认为 t3=t1。

--t4（-t4）：t4 值，默认为 t4=t2。

--overwrite（-ow）：是否覆盖上次操作的结果。

版权所有，严禁传播，违者自负法律责任！



--clustering（-cl）：是否执行聚类操作，即划分数据。

--method（-method）：默认为 mapreduce。还可选 sequential，执行单机模式。

这里使用的是余弦相似度，与 K-means 算法一致，而根据经验，0.3 已经是一个合格的阈值了，为了初始聚类中心宽泛一些，T2 值选择了 0.4，T1 值选择了 0.7。

查看 Canopy 算法生成的文件目录：

```
[root@master bin]# hadoop fs -ls /tmp/cetc55/canopy-centroids
Found 1 items
drwxr-xr-x -root supergroup 0 2017-03-08 06:14 /tmp/cetc55/canopy-centroids/clusters-0-final
```

生成了一个聚类结果，该聚类结果将直接用作初始化聚类中心

成功运行的过程中 INFO 文档如下所示。

```
17/03/08 06:13:29 INFO RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/03/08 06:13:33 INFO FileInputFormat: Total input paths to process : 1
17/03/08 06:13:33 INFO JobSubmitter: number of splits:1
17/03/08 06:13:34 INFO JobSubmitter: Submitting tokens for job: job_1487107260988_0092
17/03/08 06:13:34 INFO YarnClientImpl: Submitted application application_1487107260988_0092
17/03/08 06:13:34 INFO Job: The url to track the job: http://master:8088/proxy/application_1487107260988_0092/
17/03/08 06:13:34 INFO Job: Running job: job_1487107260988_0092
17/03/08 06:13:54 INFO Job: Job job_1487107260988_0092 running in uber mode: false
17/03/08 06:13:54 INFO Job:  map 0% reduce 0%
17/03/08 06:14:07 INFO Job:  map 100% reduce 0%
17/03/08 06:14:19 INFO Job:  map 100% reduce 100%
17/03/08 06:14:19 INFO Job: Job job_1487107260988_0092 completed successfully
17/03/08 06:14:19 INFO Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=3572
        FILE: Number of bytes written=196243
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=3458
```

版权所有，严禁传播，违者自负法律责任！



```
        HDFS: Number of bytes written=6893
        HDFS: Number of read operations=7
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Data-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=11526
        Total time spent by all reduces in occupied slots (ms)=9334
        Total time spent by all map tasks (ms)=11526
        Total time spent by all reduce tasks (ms)=9334
        Total vcore-seconds taken by all map tasks=11526
        Total vcore-seconds taken by all reduce tasks=9334
        Total megabyte-seconds taken by all map tasks=11802624
        Total megabyte-seconds taken by all reduce tasks=9558016
    Map-Reduce Framework
        Map input records=30
        Map output records=27
        Map output bytes=3501
        Map output materialized bytes=3572
        Input split bytes=132
        Combine input records=0
        Combine output records=0
        Reduce input groups=1
        Reduce shuffle bytes=3572
        Reduce input records=27
        Reduce output records=14
        Spilled Records=54
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=253
        CPU time spent (ms)=3640
        Physical memory (bytes) snapshot=330743808
        Virtual memory (bytes) snapshot=1689477120
        Total committed heap usage (bytes)=164302848
```

版权所有，严禁传播，违者自负法律责任！



```
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=3326
        File Output Format Counters
                Bytes Written=6893
    17/03/08 06:14:19 INFO MahoutDriver: Program took 54406 ms (Minutes: 0.90676
66666666667)
```

③ 运行 K-Means。只采用 tfidf-vectors 作为输入向量，并且在-c 后面用 Canopy 聚类结果。

运行命令：

```
    [root@master bin]# cd  $MAHOUT_HOME/bin
    [root@master bin]# export  MAHOUT_LOCAL=
    [root@master bin]# mahout  kmeans  -i  /tmp/cetc55/txt-sparse/tfidf-vectors
-c /tmp/cetc55/canopy-centroids/clusters-0-final  -o  /tmp/cetc55/txt-kmeans
  -dm  org.apache.mahout.common.distance.CosineDistanceMeasure  -x  100
-ow  --clustering  --cd  0.3
```

可以看到，在上述代码中并没有设定 K 值。

查看聚类结果文件目录：

```
    [root@master bin]# hadoop  fs  -ls  /tmp/cetc55/txt-kmeans
    Found 4 items
    -rw-r--r--  1 root supergroup  194 2017-03-08 06:35 /tmp/cetc55/txt-kmeans/
_policy
    drwxr-xr-x  - root supergroup  0 2017-03-08 06:36 /tmp/cetc55/txt-
kmeans/clusteredPoints
    drwxr-xr-x  - root supergroup  0 2017-03-08 06:34 /tmp/cetc55/txt-kmeans/
clusters-0
    drwxr-xr-x  - root supergroup  0 2017-03-08 06:35 /tmp/cetc55/txt-kmeans/
clusters-1-final
```

④ 查看聚类结果。将聚类结果直接给出如下文所示（操作过程中可以自行下载到 PC 端使用 Notepad++查看）：

版权所有，严禁传播，违者自负法律责任！



```
Top Terms:
        大数据开发工程师                    => 2.0033020973205566
        spark                              => 1.9756784439086914
        hadoop                             => 1.802971887588501
        hbase                              => 1.763178253173828
        hive                               => 1.6252291202545166
        mapreduce                          => 1.4128054141998292
        java                               => 1.3545177459716797
        storm                              => 1.3047189712524414
        nosql                              => 1.1167037963867188
        linux                              => 0.8465735912322998
        mysql                              => 0.839444923400879
        redis                              => 0.837527847290039
        python                             => 0.7344992160797119
        shell                              => 0.6965267658233643
        elasticsearch                      => 0.6605170249938965
Top Terms:
        Web 开发工程师                      => 2.0033020973205566
        css                                => 1.763178253173828
        ajax                               => 1.6252291202545166
        javascript                         => 1.6252291202545166
        jquery                             => 1.4731723308563232
        html5                              => 1.4731723308563232
        js                                 => 1.4128054141998292
        Web                                => 1.4023114681243896
        html                               => 1.3047189712524414
        bootstrap                          => 1.3047189712524414
        css3                               => 1.1167037963867188
        mvc                                => 0.9044709205627441
        div                                => 0.6605170249938965
        xml                                => 0.6605170249938965
        mysql                              => 0.6295836925506592
Top Terms:
        openstack                          => 2.0033020973205566
        云计算研发工程师                    => 2.0033020973205566
        kvm                                => 1.4731723308563232
        python                             => 1.4689984321594238
```

版权所有，严禁传播，违者自负法律责任！



```
linux                    =>  1.4246501922607422
虚拟化                   =>  1.3047189712524414
shell                    =>  0.9287023544311523
vmware                   =>  0.9044709205627441
sdn                      =>  0.9044709205627441
java                     =>  0.8465735912322998
c++                      =>  0.7828313827514648
ceph                     =>  0.6605170249938965
docker                   =>  0.6605170249938965
xen                      =>  0.6605170249938965
c                        =>  0.6029806137084961
```

结果同样非常的理想。

经仔细对比，发现其中关键词的排序以及权重与上一节的结果有细微不同，这正说明了两种方法过程的不同，但都得到了很好的聚类结果。

在本节中，完整阐述了模型的结构、算法的原理，并给出了命令行代码来实现，最终得出了两种方法，即 K-means 与 Canopy+K-means。前者适用于已知聚类数情况，后者适用于未知聚类数的情况。并且这两个方法都能有效地将岗位名称排序到聚类结果的前几位，从而可以直观地看出这一个聚类代表的是哪一个岗位。每一个聚类中，根据权重的排序也可以很直观地得到与该岗位关联度最大的技能点。这样的映射是来源于数以万计的大数据量岗位信息的综合评估，是客观的，是符合大数据挖掘机器学习核心理念的。

由于本节进行的是测试数据的实现，在进一步实现真实大数据分析的时候，会遇到以下问题：

笔记

1）参数会有变化。这里的参数主要指聚类的阈值，即 Canopy 算法中的 T1 与 T2，以及 K-means 算法中的收敛阈值 ConvergenceDelta，由于采用的是余弦距离，因此可以利用 0.1 为间隔，去尝试 0~1 中的值，来对比聚类结果。

2）同一种岗位会出现许多不同的名称。这是由于各个公司发布招聘信息的自主性，同时还有些公司会大量地发布相同的岗位信息，而往往这些岗位名称很长，这是公司为了提高搜索量的手段，这就需要能够合理地过滤掉这些“垃圾”招聘信息，最简单的方式可以是只保留一定长度以内岗位名称的岗位信息。

3）由于岗位名称的多样性，会使岗位名称的权重下降，可能会导致无法在聚类结果中直观地看出这一类代表的是哪一个岗位信息，因此需要人为地提高岗位名称的权重，可采用以下可行的方法。

① 让岗位名称在文本词向量中重复若干次，相当于提高词频，但是逆向词文件频率不变，提高了岗位名称的 TF-IDF 权重。

② 将同一个岗位统一化，如云计算研发工程师与云计算开发工程师其实是一个岗位，将两者都转换为同个名称，也可以大大提高岗位名称的权重。

版权所有，严禁传播，违者自负法律责任！



### 项目实训

【实训任务】

利用 Mahout 命令行方法进行 K-means 文本聚类。

【实训目的】

1. 理解文本聚类的处理流程。
2. 理解 Canopy 和 K-means 算法的机制。
3. 学会 Mahout 算法库的命令行基本操作。

【实训内容】

调整 K-means 与 Canopy 算法的收敛阈值参数，对比得到的不同聚类结果，分析区别与原因。

## 任务 5.2 使用 Mahout 的 Java API 进行岗位分析聚类开发

### 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，了解 Mahout 的源代码。
2. 学习 Mahout 的 Java API 用法。
3. 编程实现文本序列文件的向量化转换，使用 Canopy 算法初始化聚类重心，运行 K-means 算法对岗位信息进行聚类以及利用 Mahout 的 dump 接口对聚类结果进行解析。

### 任务目标

1. 熟悉 Mahout 的 Java API 的用法。
2. 学会文本序列文件的向量化转换，运行 Canopy 算法初始化聚类重心，运行 K-means 算法对岗位信息进行聚类以及利用 Mahout 的 dump 接口对聚类结果进行解析操作。

### 知识准备

微课：5-2 使用 Mahout 的 JavaAPI 进行岗位分析聚类开发

由于本任务与任务 5.1 的理论部分是一致的，即利用 K-means 进行文本聚类的流程，因此在本节中不再重复。Java 远程连接 Hadoop 集群的方法在项目 4 中也有说明。关于 Mahout 的 Java API，在 Github 上有官方的资料可以查阅，同时，*Mahout in Action* 这本书也有新闻文本聚类的代码范例，同样在 Github 也有，有兴趣的同学可以去深入了解源代码。

### 任务实施

① 利用 Mahout 自带的 sequence2TFVectors 方法将文本信息转换为<词 ID，词频>

版权所有，严禁传播，违者自负法律责任！



的向量形式，每一个词 ID 就代表了一个变量的坐标，而该词 ID 对应的词频即为该变量值，每个由词组成的岗位信息则转换为一个稀疏变量，文本序列文件便转换为了稀疏矩阵。

```
DocumentProcessor. tokenizeDocuments( new Path( inputDir) ,
                        WhitespaceAnalyzer. class, tokenizedPath, conf) ;
DictionaryVectorizer. createTermFrequencyVectors( tokenizedPath,
                        new Path( outputDir) ,
        DictionaryVectorizer. DOCUMENT_VECTOR_OUTPUT_FOLDER, conf,
                        minSupport, maxNGramSize, minLLRValue, -1. 0f, false,
                        reduceTasks, chunkSize, sequentialAccessOutput, true ) ;
}
```

这里具体的参数设置如下：

```
private final int minSupport = 2;
private final int maxNGramSize = 1;
private final float minLLRValue = 1. 0f;
private final int reduceTasks = 1;
private final int chunkSize = 64;
private final boolean sequentialAccessOutput = false;
```

这两个 Mahout 方法具体的参数含义如下：

```
public static void tokenizeDocuments(
                        Path input,
                        Class<? extends Analyzer> analyzerClass
                    Path output,
                        Configuration baseConf)
            throws   IOException,
                        InterruptedException, ClassNotFoundException
```

参数说明：

Input：输入的序列文件路径。

analyzerClass：使用的 Lucene 解析器（本案例使用 WhiteSpaceAnalyzer）。

Output：输出目录。

baseConf：Hadoop 配置。

```
public static void createTermFrequencyVectors(
                        Path input,
                        Path output,
                        String tfVectorsFolderName,
```

版权所有，严禁传播，违者自负法律责任！



```
                                    Configuration baseConf,
                                    int minSupport,
                                    int maxNGramSize,
                                    float minLLRValue,
                                    float normPower,
                                    boolean logNormalize,
                                    int numReducers,
                                    int chunkSizeInMegabytes,
                                    boolean sequentialAccess,
                                    boolean namedVectors)
                            throws  IOException,
                                        InterruptedException,
                                    ClassNotFoundException
```

参数说明：

Input：输入词条化后的文件目录。

Outout：输出目录。

tfVectorsFolderName：词频向量保存的文件名。

baseConf：Hadoop 配置。

minSupport：考虑加入向量的最低词频（在语料库中）。

maxNGramSize：最大的连词大小（这里采用 1）。

minLLRValue：考虑删除连词的最小对数似然比。

normPower：计算的 L_p norm。

logNormalize：考虑是否使用对数正态化。

numReducers：使用的 reducer 数目。

chunkSizeInMegabytes：在每个节点保存的序列文件的大小。

这里的中文文本是由以空格（制表符）分隔开的，因此可以当成英文文本来处理，所有的参数设置使用 default（默认）即可，具体可见 Github 的 Mahout 源代码。

在上一节中使用了 TF-IDF 值作为文本向量化的基础，但在本节中，考虑到只保留了有用的词汇（岗位名称与技能点），因此使用 TF 值是最为合适的。

② 使用 Canopy 算法 API 对向量化后的文件初始化聚类中心，具体代码如下：

```
    // 运行 Canopy 算法生成初始聚类中心,调用 Mahout 算法库封装好的 CanopyDriver 方法
    CanopyDriver.run(conf, vectorsFolder, canopyCentroids,
                                    new CosineDistanceMeasure(), 0.6, 0.4, false, 0.0,
false);
```

需要设置的 3 个重要参数分别为距离选择，以及阈值 T1、T2 的取值。这里选择的是适合文本分析的余弦距离公式，以及 T1 为 0.6，T2 为 0.4。

CanopyDriver. run 方法如下：

```
public static void run(Configuration conf,
                       Path input,
                       Path output,
                            DistanceMeasure measure,
                                    double t1,
                                    double t2,
                            boolean runClustering,
                                    double clusterClassificationThreshold,
                                    boolean runSequential)
                        throws   IOException,
                                        InterruptedException,
                                    ClassNotFoundException
```

参数说明：

conf：Hadoop 配置。

Input：输入的向量路径。

Output：输出的初始聚类中心。

Measure：选择的距离计算公式。

t1：T1，距离度量。

t2：T2，距离度量。

runClustering：是否要聚类输入向量 。

clusterClassificationThreshold：概率分布函数低于该阈值的向量将不被聚类。该阈值取值为 0 到 1。

runSequential：表示在本地执行而不是在 MapReduce 中执行。

③ 在运行完 Canopy 方法后，便要进行 K-means 算法，利用 Canopy 生成的结果作为初始聚类中心，由于此时聚类中心数已经确定，不需要设置 K 值，最终的聚类数会根据收敛阈值变化而变化。

具体代码如下：

```
//运行 K-means 算法得到聚类结果,调用 Mahout 算法库中封装好的 KmeansDriver 方法,参数给定
            KMeansDriver. run(conf, vectorsFolder, new Path(canopyCentroids,
                        Constants. CLUSTERS _ 0 _ FINAL ), clusterOutput, 0. 3,
100, true, 0. 0,
                        false );
    }
```

其中重要的参数设置为收敛阈值为 0. 3，迭代次数为 100。

KmeansDriver. run 方法如下：

```
public static void run(Configuration conf,
                       Path input,
                       Path clusterIn
                     Path output,
                           Double convergenceDelta,
                                  int maxIterations,
                           boolean runClustering,
                                  double clusterClassificationThreshold,
                                  boolean runSequential)
                            throws   IOException,
                           InterruptedException,
                           ClassNotFoundException
```

④ 调用 Mahout 的 ClusterDumper 类将最终的聚类结果转换为可读的信息。由于聚类结果是序列文件，而且是以词 ID 的形式出现，因此需要将在文本向量化过程中生成的 dictionary 文件关联至该方法。具体实现的代码如下：

```
ClusterDumper clusterDumper=new ClusterDumper();
   String[ ]args={ "--input",
   Constants. KMEANS _ OUTPUTDIR +"/" + Constants. CLUSTERS +"/" + Constants. CLUSTERS_3_FINAL, "--output", dumppath,
   "-d",  Constants. KMEANS_OUTPUTDIR +"/" +Constants. DICTIONARY,
   "-p",
Constants. KMEANS_OUTPUTDIR +"/" +Constants. CLUSTERS +"/" +Constants. CLUSTEREDPOINTS,
   "-dt", Constants. SEQUENCEFILE, "-n", Constants. DUMP_NUM };
                    int result=clusterDumper. run(args);
```

最后得到的聚类文件形式如图 5-3 所示。

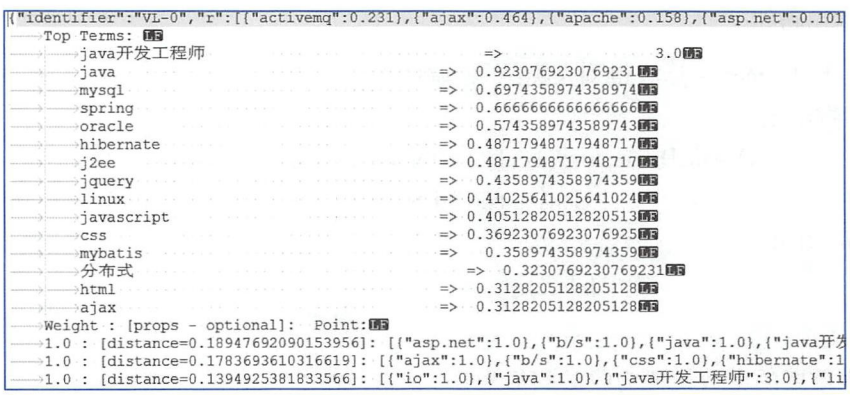



图 5-3 聚类文件展示

## 项目实训

【实训任务】

利用 Mahout 的 Java API 进行 K-means 文本聚类开发。

【实训目的】

1. 理解文本聚类的处理流程。
2. 理解 Canopy 和 K-means 算法的机制。
3. 学会 Mahout 相关算法功能的 Java API 开发。

【实训内容】

调整 K-means 与 Canopy 算法的收敛阈值参数，对比得到的不同聚类结果，总结区别与原因。调整文本词向量转换过程中的参数设置，理解默认参数的含义，以及修改参数对最终聚类结果的影响。

# 任务 5.3　使用 Mahout 推荐工作岗位

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习皮尔森相关度、欧式距离相似度和余弦相似度、Spearman 秩相关系数、曼哈顿距离、对数似然相似度等相关知识。

2. 基于上述岗位聚类分析结果和学生的成绩信息，实现对学生的岗位推荐。

## 任务目标

1. 学会将成绩信息与岗位技能权重读入数组保存的操作。
2. 学会将数组写入 HDFS 向量序列文件的操作。
3. 学会运行 Mahout 计算向量相似度算法的操作。
4. 学会通过解析算法结果实现岗位推荐的操作。

## 知识准备

这里使用的是推荐算法，即通过学生的技能成绩与岗位分析得到的技能权重，进行相似度匹配，其本质为向量的相似度算法。以下是 Mahout 算法库中已有的相似度算法。

（1）皮尔森相关度

类名：Pearson Correlation Similarity。

原理：用来反映两个变量线性相关程度的统计量。

范围：[-1，1]，绝对值越大，说明相关性越强，负相关对于推荐的意义小。

说明：

笔记

① 不考虑重叠的数量。

② 如果只有一项重叠，无法计算相似性（计算过程被除数有 n-1）。

③ 如果重叠的值都相等，也无法计算相似性（标准差为 0，做除数）。

该相似度并不是最好的选择，也不是最坏的选择，只是因为其容易理解，在早期研究中经常被提起。使用 Pearson 线性相关系数必须假设数据是成对地从正态分布中取得的，并且数据至少在逻辑范畴内必须是等间距的数据。Mahout 中，为皮尔森相关计算提供了一个扩展，通过增加一个枚举类型（Weighting）的参数来使得重叠数也成为计算相似度的影响因子。

（2）欧式距离相似度

类名：Euclidean Distance Similarity。

原理：利用欧式距离 d 定义的相似度 s，s=1/(1+d)。

范围：[0，1]，值越大，说明 d 越小，也就是距离越近，则相似度越大。

说明：同皮尔森相似度一样，该相似度也没有考虑重叠数对结果的影响，同样地，Mahout 通过增加一个枚举类型（Weighting）的参数来使得重叠数也成为计算相似度的影响因子。

（3）余弦相似度

类名：Pearson Correlation Similarity 和 Uncentered Cosine Similarity。

原理：多维空间两点与所设定的点形成夹角的余弦值。

范围：[-1，1]，值越大，说明夹角越大，两点相距就越远，相似度就越小。

说明：在数学表达中，如果对两个项的属性进行了数据中心化，计算出来的余弦相似度和皮尔森相似度是一样的，在 Mahout 中，实现了数据中心化的过程，所以皮尔森相似度值也是数据中心化后的余弦相似度。在新版本中，Mahout 提供了 Uncentered Cosine Similarity 类作为计算非中心化数据的余弦相似度。

（4）Spearman 秩相关系数

类名：Spearman Correlation Similarity。

原理：Spearman 秩相关系数通常被认为是排列后的变量之间的 Pearson 线性相关系数。

范围：{-1.0，1.0}，当一致时为 1.0，不一致时为-1.0。

说明：计算非常慢，有大量排序。针对推荐系统中的数据集来讲，用 Spearman 秩相关系数作为相似度量是不合适的。

（5）曼哈顿距离

类名：City Block Similarity

原理：曼哈顿距离的实现，同欧式距离相似，都是用于多维数据空间距离的测度。

范围：[0，1]，同欧式距离一致，值越小，说明距离值越大，相似度越大。

说明：比欧式距离计算量少，性能相对高。

（6）Tanimoto 系数

类名：Tanimoto Coefficient Similarity

原理：又名广义 Jaccard 系数，是对 Jaccard 系数的扩展。

范围：[0，1]，完全重叠时为 1，无重叠项时为 0，越接近 1 说明越相似。

说明：处理无打分的偏好数据。

（7）对数似然相似度

类名：Log Likelihood Similarity

原理：重叠的个数，不重叠的个数，都没有的个数。

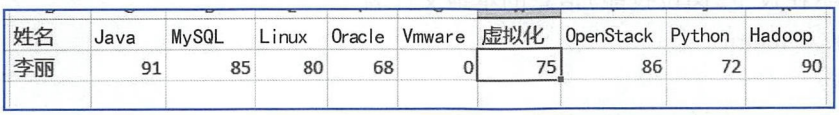
范围：具体可参考百度文库中论文 *Accurate Methods for the Statistics of Surprise and Coincidence*。

说明：处理无打分的偏好数据，比 Tanimoto 系数的计算方法更为智能。

在本案例中使用的是常规的欧氏距离相似度。

## 任务实施

① 将成绩信息与技能权重读入数组保存。成绩信息如图 5-4 所示。



| 姓名 | Java | MySQL | Linux | Oracle | Vmware | 虚拟化 | OpenStack | Python | Hadoop |
|---|---|---|---|---|---|---|---|---|---|
| 李丽 | 91 | 85 | 80 | 68 | 0 | 75 | 86 | 72 | 90 |

图 5-4　成绩信息表

微课：5-3 使用 Mahout 推荐工作岗位

岗位聚类结果通过持久化至数据库，导出至 Excel 表格，具体是如图 5-5 所示。

| Jobname | Java | MySQL | Linux | Oracle | Vmware | 虚拟化 | OpenStack | F |
|---|---|---|---|---|---|---|---|---|
| Java开发工程师 | 0.92 | 0.7 | 0.41 | 0.57 | 0 | 0 | 0 | |
| 系统集成工程师 | 0 | 0.3 | 0.7 | 0.8 | 0.8 | 0.6 | 0 | |
| 系统运维工程师 | 0 | 0.65 | 0.94 | 0 | 0 | 0.65 | 0.53 | |
| 系统架构工程师 | 0.74 | 0.55 | 0.42 | 0.66 | 0 | 0.37 | 0 | |
| 系统开发工程师 | 0.3 | 0.34 | 0.78 | 0 | 0.34 | 0.56 | 0 | |
| 数据库开发工程师 | 0.43 | 0.43 | 0.71 | 0.57 | 0 | 0 | 0 | |
| 大数据架构工程师 | 0.81 | 0.33 | 0.48 | 0.29 | 0 | 0 | 0 | |
| 大数据开发工程师 | 0.76 | 0.32 | 0.53 | 0 | 0 | 0 | 0 | |
| 云计算运维工程师 | 0.33 | 0.46 | 0.85 | 0 | 0 | 0.63 | 0.5 | |
| 云计算架构工程师 | 0.43 | 0.38 | 0.55 | 0 | 0 | 0.52 | 0.62 | |
| 云计算开发工程师 | 0.46 | 0 | 0.71 | 0 | 0 | 0.46 | 0.44 | |
| Web开发工程师 | 0.42 | 0 | 0 | 0.21 | 0 | 0 | 0 | |

图 5-5　岗位聚类结果信息表

这里由于成绩与权重的取值范围不同，因此需要将其统一化。选择将成绩映射至[0，1] 区间，选择的映射公式如下：

$$统一化后的成绩=\frac{原始成绩-50}{50}$$

将岗位技能信息以及规范化后的学生成绩信息保存至 CSV 文件（或以逗号隔开的 txt 文本文件）。将其保存至数组，具体代码如下：

```
//将 CSV 读取,保存在 String 数组中
String[ ][ ] JobSkill = csvRead( jobSkill) ;
String[ ][ ] SchoolStu = csvRead( schoolStu) ;

//读取其中毕业生与在校生成绩信息
double[ ][ ] SkillWeight = stringgetData( JobSkill) ;
double[ ][ ] SchoolScore = stringgetData( SchoolStu) ;
```

其中用到了两个自定义的方法，代码如下：

```
/**
   * 将 CSV 文件读取到二维 String 数组
   *
   * @ param filename
   * :在校生(岗位技能)信息的本地文件名
   * @ return
   */
private String[ ][ ] csvRead( String filename) {
        List<String[ ]> list = new ArrayList<String[ ]>( ) ;
        BufferedReader reader = null;
        try {
            reader = new BufferedReader( new InputStreamReader(
                        new FileInputStream( filename) , "utf-8" ) ) ;
            String line = null;

            while ( ( line = reader. readLine( ) ) ! = null) {
                String item[ ] = line. split( " ," ) ;
                list. add( item) ;
            }
        } catch ( Exception e) {
            e. printStackTrace( ) ;
        } finally {
            try {
                reader. close( ) ;
            } catch ( Exception e) {
                e. printStackTrace( ) ;
            }
        }
```

```
        String[][] dataframe=new String[list.size()][];
        for (int i=0; i < dataframe.length; i++) {
            String[] row=list.get(i);
            dataframe[i]=row;
        }
        return dataframe;
    }
    /**
     *将二维 String 数组中的成绩信息读到二维 double 数组
     *
     * @param dataframe
     *:在校生(岗位技能)的信息二维数组
     * @return
     */
    private double[][] stringgetData(String[][] dataframe) {
        double[][] data=new double[dataframe.length-1][dataframe[0].length-1];
        for (int i=0; i < data.length; i++) {
            for (int j=0; j < data[0].length; j++) {
                double tmp=Double.parseDouble(dataframe[i+1][j+1]);
                data[i][j]=tmp;
            }
        }
        return data;
    }
```

② 将数组写入 HDFS 向量序列文件。接着，便是将两组成绩数据写入 HDFS 的序列向量文件。

```
//定义配置文件以及输入输出路径
        Configuration config=new Configuration();
        config.set("fs.defaultFS","hdfs://192.168.56.102:8020");
        FileSystem fs=FileSystem.get(config);
        Path path1= new Path(Constants.RECOMMJOB_INPUT);
        Path path2= new Path(Constants.RECOMMJOB_SEEDS);
        Path path3= new Path(Constants.RECOMMJOB_OUTPUT);
        HadoopUtil.delete(config, path1);
        HadoopUtil.delete(config, path2);
```

```
                HadoopUtil.delete(config, path3);

                // 将数组形式的成绩写入 HDFS 的序列文件
                List<VectorWritable> points = getPointsWritable(JobSkill);
                List<VectorWritable> seeds = getPointsWritable(SchoolScore);
                writePointsToFile(points, true, path1, fs, config);
        writePointsToFile(seeds, true, path2, fs, config);
```

定义配置文件的代码部分不再赘述，写入数据的代码用到了以下两个方法，代码如下：

```
/**
 * 将二维数组转成 HDFS 上可以识别的 vectors
 *
 * @param raw
 *          :在校生(岗位技能)的成绩数组
 * @return
 */
private List<VectorWritable> getPointsWritable(double[][] raw) {
        List<VectorWritable> points = Lists.newArrayList();
        for (double[] fr : raw) {
                RandomAccessSparseVector vec = new RandomAccessSparseVector(
fr.length);
                vec.assign(fr);
                points.add(new VectorWritable(vec));
        }
        return points;
}
/**
 * 将生成的 vectors 写入 HDFS 可以运行的序列文件
 *
 * @param points
 *          :在校生(岗位技能)的成绩转成的 vectors
 * @param intWritable
 *          : Key 是否为 IntWritable(在此案例中为 true)
 * @param path
 *          :在 HDFS 输出的序列文件的目录
 * @param fs
```

```
     *          : HDFS 文件系统
     * @param conf
     *          :HDFS 配置
     * @throws IOException
     */
    private void writePointsToFile(Iterable<VectorWritable> points,
                boolean intWritable, Path path, FileSystem fs, Configuration conf)
                throws IOException {
        SequenceFile.Writer writer=new SequenceFile.Writer(fs, conf, path,
                intWritable ? IntWritable.class : LongWritable.class,
                VectorWritable.class);
        try {
            int recNum=0;
            for (VectorWritable point : points) {
                writer.append(intWritable ? new IntWritable(recNum++)
                        : new LongWritable(recNum++), point);
            }
        } finally {
            Closeables.close(writer, false);
        }
    }
```

③ 运行 Mahout 计算向量相似度算法。其代码非常简单，只有一行，该行也是整个 Java 程序的核心所在。

```
VectorDistanceSimilarityJob.run(config,path1, path2, path3, new EuclideanDistance-
Measure(), "pw");
```

这里使用的是欧氏距离算法。

④ 解析相似度结果并推荐岗位。运行完步骤③后，生成了一个序列文件保存结果，该序列文件的 key 为进行比较的两组向量的 ID 信息，value 为这两组向量的距离。只需要通过 SequenceFile.Reader 将距离信息读出，然后进行排序，最后结合毕业生的成绩信息和岗位信息文件，做出推荐，最后将具体的推荐结果写入一个新的文件中，具体代码如下：

```
//读取相似度结果文件并保存至数组
double[ ] Dist=new double[ Grad_Num];
int count=0;
SequenceFile.Reader reader = new SequenceFile.Reader(fs, new Path(Constants.
RECOMMJOB_OUTPUT,Constants.RECOMMJOB_DIST_RESULT), config);
```

非卖品，仅供非商业用途或交流学习使用

```
StringTuple key=new StringTuple();
DoubleWritable value=new DoubleWritable();
while (reader.next(key, value)) {
   Dist[count]=Double.parseDouble(value.toString());
   count++;
}
reader.close();
//对数组进行排序
double[][] order=sortingResult(Dist);
//推荐用户
String uname=SchoolStu[1][0];
List list=new ArrayList();
list.add(uname);
//推荐前四个岗位
String jobs="";
BufferedWriter out=null;
out=new BufferedWriter(new OutputStreamWriter(
        new FileOutputStream(resultJob, true)));
out.write("为"+uname+"推荐岗位为:"+"\r\n");
for (int i=0; i<4; i++) {
   String job=GradStu[(int) order[i][1]+1][0];
   list.add(job);
   out.write(job+"\r\n");
}
out.close();
```

## 项目实训

**【实训任务】**

利用 Mahout 的 Java API 进行推荐算法的实现。

**【实训目的】**

1. 熟悉基于用户的推荐算法机制。
2. 熟悉各个向量相似度的区别以及适用的场景。
3. 学会将向量数据写入 Mahout 序列文件的方法。

**【实训内容】**

使用多个向量相似度进行推荐算法的实现，比较分析存在的异同点。

非卖品，仅供非商业用途或交流学习使用

## 项目小结

本项目主要介绍了对岗位数据的数据分析。数据分析是大数据挖掘流程中最为重要的一步，涉及抽象的数学模型与机器学习算法，需要的理论基础较多。本项目中岗位分析主要使用了文本聚类的模型，其核心为 K-means 算法，使用了 Mahout 机器学习算法库。基于 Mahout 封装的 Java API，可以用轻便的代码量实现复杂的算法模型。在岗位推荐服务中，使用了向量的相似度算法，同样也是采用 Mahout 算法库中的方法。其中有许多可以自定义的地方，如文本向量处理部分的参数设置、Canopy 与 K-means 算法的收敛阈值参数设置，以及向量相似度算法的距离度量选择，都是值得理解与研究的地方。

非卖品，仅供非商业用途或交流学习使用

# 项目 6

# 数据的可视化

PPT6：
数据的可视化

## 学习目标

【知识目标】

- 掌握数据可视化流程。
- 了解数据可视化的图形选择。

【技能目标】

- 学会使用 Excel 进行数据可视化操作。
- 学会使用 ECharts 进行数据可视化操作。
- 学会使用 D3. js 进行数据可视化操作。

## 学习情境

米老师在完成了数据采集、数据清洗和数据分析（挖掘）后，希望可以将分析结果直观地展现出来，因此需要借助数据可视化工具来进行实现。现有许多软件工具（语言）是集数据分析与数据可视化为一体的，如 RStudio、MATLAB 和 Mathematica，但是这些软件工具（语言）更多的是适用于格式化向量化的数据，图形种类也不够丰富。最终，米老师选择分别采用电子表格（Excel）、ECharts 和 D3. js 来进行数据可视化的实现。

非卖品，仅供非商业用途或交流学习使用

# 任务 6.1 使用 Excel 可视化数据

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习使用 Excel 可视化数据的操作。

2. 使用 Excel，完成岗位技能点数据、岗位变化的数据和岗位数量的数据可视化操作。

## 任务目标

1. 学会使用 Excel 的雷达图对岗位技能点数据做可视化展示操作。

2. 学会使用 Excel 的折线图对岗位需求量数据做可视化展示操作。

## 知识准备

Microsoft Excel 是微软公司的办公软件 Microsoft Office 的组件之一，是由微软公司为 Windows 和 Apple Macintosh 操作系统的计算机而编写和运行的一款制表软件。Excel 是微软办公套装软件的一个重要的组成部分，它可以进行各种数据的处理、统计分析和辅助决策操作，广泛地应用于管理、统计财经、金融等众多领域。

Excel 中有大量的公式函数可以在应用中进行选择，使用 Microsoft Excel 可以执行计算、分析信息并管理电子表格或网页中的数据信息列表，并进行数据资料图表制作，可以实现许多方便的功能。

## 任务实施

微课：6-1 使用 Excel 可视化数据

① 根据分析的岗位技能点数据，制作 Excel 表格，如图 6-1 所示。

| | Hadoop | Hbase | Linux | Hive | 分布式 | Spark |
|---|---|---|---|---|---|---|
| 大数据开发工程师 | 0.769231 | 0.615385 | 0.846154 | 0.615385 | 0.769231 | 0.692308 |
| 大数据架构工程师 | 0.95122 | 0.609756 | 0.341463 | 0.658537 | 0.658537 | 0.902439 |
| 大数据开发工程师 | 0.785235 | 0.47651 | 0.483221 | 0.483221 | 0.496644 | 0.510067 |

图 6-1 岗位技能点信息表

② 选中所有的数据，单击主菜单栏中“插入”按钮，在弹出的下拉菜单中选择“图表”→“雷达图”菜单命令，生成图表，如图 6-2 所示。

也可以根据岗位变化的数据，制作 Excel 表格，如图 6-3 所示。

选中所有的数据，单击主菜单栏中“插入”按钮，在弹出的下拉菜单中选择“图表”→“折线图”→“堆积折线图”命令，生成图表，如图 6-4 所示。

同时可以根据岗位数量的数据，制作 Excel 表格，如图 6-5 所示。

非卖品，仅供非商业用途或交流学习使用

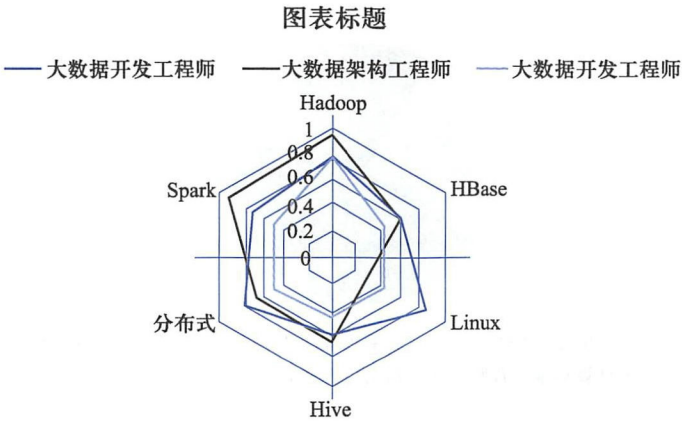



图 6-2　岗位技能点的雷达图

| | 3月21日 | 3月22日 | 3月23日 | 3月24日 | 3月25日 | 3月26日 | 3月27日 |
|---|---|---|---|---|---|---|---|
| Java开发工程师 | 4 | 2 | 2 | 6 | 4 | 4 | 6 |
| 云计算开发工程师 | 4 | 8 | 14 | 6 | 2 | 2 | 6 |
| 云计算架构工程师 | 2 | 2 | 4 | 2 | 6 | 2 | 2 |
| 云计算运维工程师 | 2 | 4 | 2 | 2 | 2 | 2 | 2 |
| 系统开发工程师 | 2 | 2 | 2 | 2 | 3 | 2 | 2 |

图 6-3　岗位变化信息表

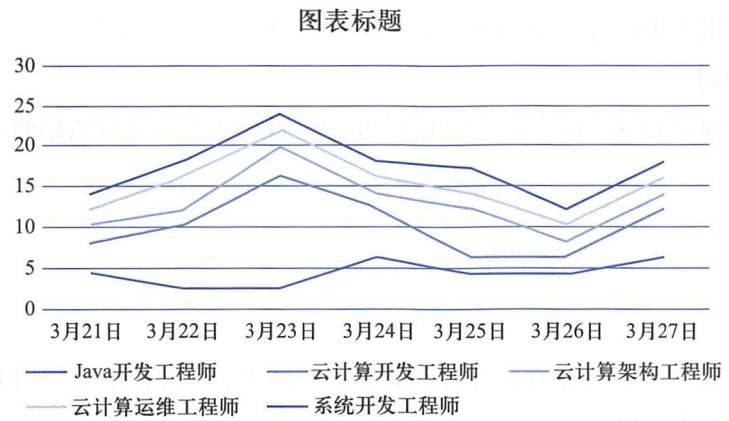



图 6-4　岗位变化的堆积折线图

| | 数量 |
|---|---|
| Java开发工程师 | 2 |
| 云计算开发工程师 | 14 |
| 云计算架构工程师 | 4 |
| 云计算运维工程师 | 2 |
| 系统开发工程师 | 2 |

图 6-5　岗位数量信息表

选中所有的数据，单击主菜单栏中“插入”按钮，在弹出的下拉菜单中选择“图表”→“饼图”命令，生成图表，如图 6-6 所示。

非卖品，仅供非商业用途或交流学习使用

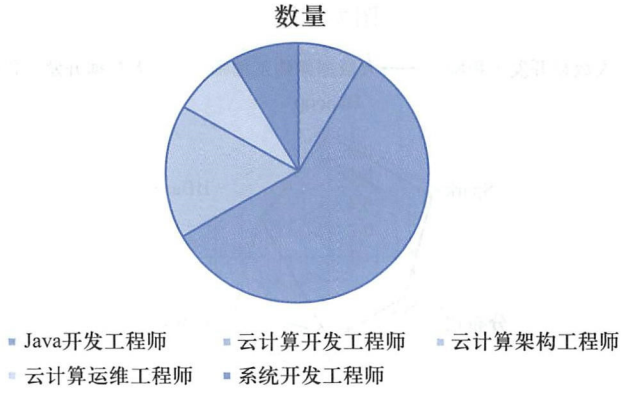



图 6-6 岗位数量的饼图

## 项目实训

【实训任务】

利用 Excel 完成数据可视化操作。

【实训目的】

1. 熟悉 Excel 各个数据图的使用操作。
2. 根据具体应用需求，学会 Excel 图表样式的选择与使用操作。

【实训内容】

将任务实施中的各个操作，均选用不同的 Excel 图表样式进行数据的可视化操作。

# 任务 6.2 使用 ECharts 可视化数据

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习使用 ECharts 可视化数据的操作相关知识。

2. 使用 ECharts，完成岗位需求变化、岗位分类、岗位技能点的和岗位地区分布的数据可视化操作。

## 任务目标

1. 学会使用 ECharts 实现岗位需求变化趋势的可视化操作。
2. 学会使用 ECharts 实现岗位分类的可视化操作。
3. 学会使用 ECharts 实现岗位技能点的可视化操作。
4. 学会使用 ECharts 实现岗位地区分布的可视化操作。

非卖品，仅供非商业用途或交流学习使用

## 知识准备

ECharts，一个纯 JavaScript 的图表库，可以流畅地运行在个人计算机和移动设备上，兼容当前绝大部分浏览器（如 IE 8/9/10/11、Chrome、Firefox、Safari 等），底层依赖轻量级的 Canvas 类库 ZRender，提供直观、生动、可交互、可高度个性化定制的数据可视化图表。

ECharts 提供了常规的折线图、柱状图、散点图、饼图、K 线图、用于统计的盒形图、用于地理数据可视化的地图、热力图、线图、用于关系数据可视化的关系图、treemap、多维数据可视化的平行坐标，还有用于 BI 的漏斗图、仪表盘，并且支持图与图之间的混搭。

## 任务实施

微课：6-2 使用 ECharts 可视化数据

① 在 ECharts 官网（http://ECharts.baidu.com/download.html）下载 ECharts 3.0 的 js 文件。下载完成后将 echars.min.js 文件放到项目的 assets 目录内。本案例需要展示中国地图，还需要在官网地图下载 china.json 文件，并放入到项目的 assets 目录内。Echars 3.0 是基于 jquery 可高度个性化定制的数据可视化图表，还需要下载 jquery-3.1.1.min.js 文件，并放到项目的 assets 目录内。结构如图 6-7 所示。

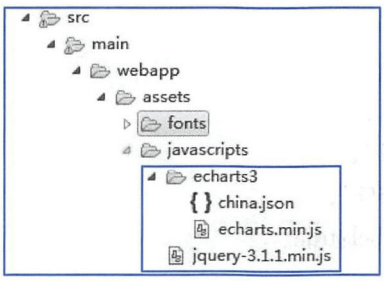



图 6-7　Echars 3.0 的文件结构

② 展示岗位数量的变化曲线图。新建一个 HTML 文件。引入 jquery 和 ECharts 的 js 文件。

```
<script type="text/javascript"
    src="${request.contextPath}/assets/javascripts/jquery-3.1.1.min.js"></script>
<script type="text/javascript"
    src="${request.contextPath}/assets/javascripts/ECharts3/ECharts.min.js"></script>
```

在 body 元素内为 ECharts 准备一个 DOM，并设定其大小。

```
<div id="main" style="width:100%"></div>
```

以下为 JavaScript 内容。其对展示的图形进行调整，使图像全屏显示。

非卖品，仅供非商业用途或交流学习使用

```
var height = window. screen. height;
height -= 120;
$("#main"). css("height",height)
```

基于准备好的 DOM，初始化 ECharts 实例。

```
var myChart = ECharts. init( document. getElementById('main'));
```

设置岗位变化数据。

```
option = {
        title: {
            text: '岗位变化曲线'
        },
        tooltip: {
            trigger: 'axis'
        },
        legend: {
            data: ['Java 开发工程师','云计算开发工程师','云计算架构工程师','云计算运维工程师','系统开发工程师']
        },
        grid: {
            left: '3%',
            right: '4%',
            bottom: '3%',
            containLabel: true
        },
        toolbox: {
            feature: {
                saveAsImage: {}
            }
        },
        xAxis: {
            type: 'category',
            boundaryGap: false,
            data: ['03-21','03-22','03-23','03-24','03-25','03-26','03-27']
        },
        yAxis: {
            type: 'value'
        },
```

非卖品，仅供非商业用途或交流学习使用

```
        series:[
            {
                name:'Java 开发工程师',
                type:'line',
                stack:'总量',
                data:[4,2,2,6,4,4,6]
            },
            {
                name:'云计算开发工程师',
                type:'line',
                stack:'总量',
                data:[4,8,14,6,2,2,6]
            },
            {
                name:'云计算架构工程师',
                type:'line',
                stack:'总量',
                data:[2,2,4,2,6,2,2]
            },
            {
                name:'云计算运维工程师',
                type:'line',
                stack:'总量',
                data:[2,4,2,2,2,2,2]
            },
            {
                name:'系统开发工程师',
                type:'line',
                stack:'总量',
                data:[2,2,2,2,3,2,2]
            }
        ]
    };
```

将数据装载到图形，进而可视化展示，如图6-8所示。

```
myChart.setOption(option);
```

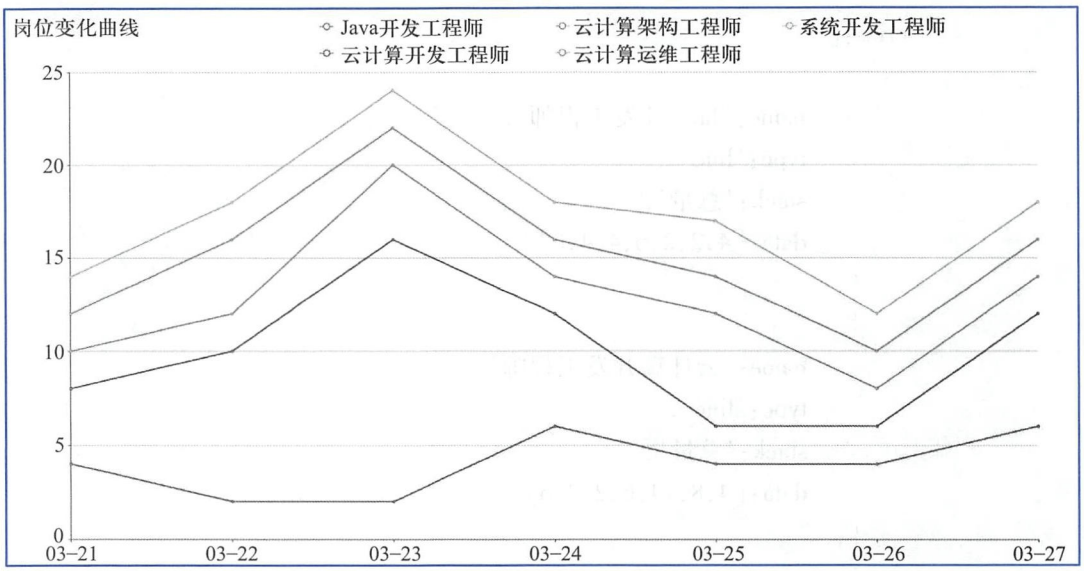



图 6-8 岗位变化的可视化

③ 展示岗位的数量图。所需要的配置不变，将岗位变化曲线图的数据（option 变量）修改为以下内容。

```
option = {
    tooltip: {
        trigger: 'item',
        formatter: "{a} <br/>{b}: {c} ({d}%)"
    },
    legend: {
        orient: 'vertical',
        x: 'left',
        data: ['Java 开发工程师','云计算开发工程师','云计算架构工程师',
'云计算运维工程师','系统开发工程师']
    },
    series: [
        {
            name: '访问来源',
            type: 'pie',
            radius: ['50%', '70%'],
            avoidLabelOverlap: false,
            label: {
                normal: {
                    show: false,
```

```
                position:'center'
            },
            emphasis:{
                show:true,
                textStyle:{
                    fontSize:'30',
                    fontWeight:'bold'
                }
            }
        },
        labelLine:{
            normal:{
                show:false
            }
        },
        data:[
            {value:2,name:'Java 开发工程师'},
            {value:14,name:'云计算开发工程师'},
            {value:4,name:'云计算架构工程师'},
            {value:2,name:'云计算运维工程师'},
            {value:2,name:'系统开发工程师'}
        ]
    }
]
};
```

数据可视化的效果如图 6-9 所示。

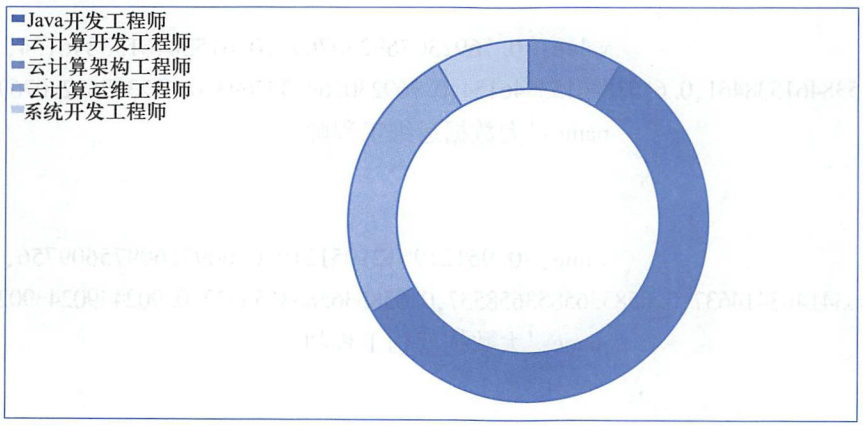



图 6-9　岗位数量的可视化

④ 展示岗位技能点的雷达图。同样所需配置保持不变，只需要将数据（option 变量）修改为以下内容。

```
option = {
        title: {
            text: '岗位技能点系数图'
        },
        tooltip: {},
        legend: {
            data: ['大数据运维工程师','大数据架构工程师','大数据开发工程师']
        },
        radar: {
            // shape: 'circle',
            indicator: [
                { name: 'Hadoop', max: 1},
                { name: 'Hbase', max: 1},
                { name: 'Linux', max: 1},
                { name: 'Hive', max: 1},
                { name: '分布式', max: 1},
                { name: 'Spark', max: 1},
            ]
        },
        series: [{
            name: 'unknow',
            type: 'radar',
            // areaStyle: {normal: {}},
            data: [
                {
                    value: [0.7692307692307693, 0.6153846153846154,
0.8461538461538461, 0.6153846153846154, 0.7692307692307693, 0.6923076923076923],
                    name: '大数据运维工程师'
                },
                {
                    value: [0.9512195121951219, 0.6097560975609756,
0.34146341463414637, 0.6585365853658537, 0.6585365853658537, 0.9024390243902439],
                    name: '大数据架构工程师'
                },
                {
```

```
                value:[0.785234899328859,0.47651006711409394,
0.48322147651006714,0.48322147651006714,0.4966442953020134,0.5100671140939598],
                name:'大数据开发工程师'
            }
        ]
    }]
};
```

数据可视化的效果如图6-10所示。

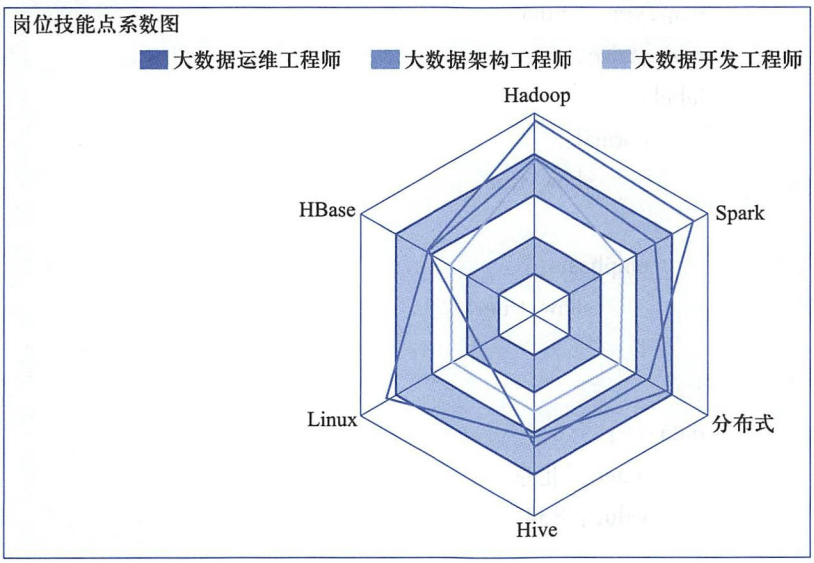



图6-10　岗位技能点的可视化

⑤ 岗位的分布图。

引入jquery和ECharts的js文件，在HTML部分同样先定义一个DOM。

```
<body style="">
    <!--为 ECharts 准备一个具备大小(宽高)的 DOM -->
    <div id="main" style="width:100%;"></div>


</body>
<script type="text/javascript"
    src="${request.contextPath}/assets/javascripts/jquery-3.1.1.min.js"></script>
<script type="text/javascript"
    src="${request.contextPath}/assets/javascripts/ECharts3/ECharts.min.js"></script>
```

JavaScript 部分调整图像展示的大小。

```
var height = window. screen. height;
    height  -= 140;
     $("#main"). css("height",height)
```

写入需要展示的岗位分布数据。

```
var data = [  {
                name:'岗位',
                type:'map',
                mapType:'china',
                roam:false,
                label:{
                    normal:{
                        show:true
                    },
                    emphasis:{
                        show:true
                    }
                },
                data:[  {
                    name:'北京',
                    value:'834'
                },{
                    name:'上海',
                    value:'499'
                },{
                    name:'天津',
                    value:'76'
                },{
                    name:'重庆',
                    value:'81'
                },{
                    name:'云南',
                    value:'105'
                },{
                    name:'广东',
                    value:'658'
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
},{
    name:'江苏',
    value:'729'
},{
    name:'浙江',
    value:'354'
},{
    name:'湖北',
    value:'269'
},{
    name:'辽宁',
    value:'149'
},{
    name:'山东',
    value:'526'
},{
    name:'四川',
    value:'310'
},{
    name:'江西',
    value:'115'
},{
    name:'陕西',
    value:'144'
},{
    name:'黑龙江',
    value:'89'
},{
    name:'广西',
    value:'101'
},{
    name:'福建',
    value:'180'
},{
    name:'贵州',
    value:'144'
},{
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
            name:'河南',
            value:'191'
        },{
            name:'内蒙古',
            value:'130'
        },{
            name:'河北',
            value:'176'
        },{
            name:'吉林',
            value:'25'
        },{
            name:'安徽',
            value:'139'
        },{
            name:'海南',
            value:'66'
        },{
            name:'山西',
            value:'221'
        },{
            name:'湖南',
            value:'803'
        },{
            name:'宁夏',
            value:'61'
        },{
            name:'甘肃',
            value:'60'
        },{
            name:'青海',
            value:'29'
        } ]
    } ];
    data=eval(data)
```

基于准备好的 DOM，初始化 ECharts 实例。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
$.get('${request.contextPath}/assets/javascripts/ECharts3/china.json',
                    function (chinaJson) {

                        ECharts.registerMap('china',chinaJson);
                        var chart =
ECharts.init(document.getElementById('main'));
                        chart.setOption({
                            title:{
                                text:"岗位分布图",
                                subtext:"江苏一道云科技发展有限公司",
                                left:'center'
                            },
                            tooltip:{
                                trigger:'item'
                            },
                            legend:{
                                orient:'vertical',
                                left:'left',
                                data:['岗位']
                            },
                            visualMap:{
                                min:0,
                                max:1000,
                                left:'left',
                                top:'bottom',
                                text:['高','低'],// 文本,默认为数值文本
                                calculable:true
                            },
                            toolbox:{
                                show:true,
                                orient:'vertical',
                                left:'right',
                                top:'center',
                                feature:{
                                    dataView:{
                                        readOnly:false
                                    },
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
                restore:{},
                saveAsImage:{}
            }
        },
        series:data
    })
})
```

## 项目实训

**【实训任务】**

利用 ECharts 实现数据的可视化操作。

**【实训目的】**

1. 熟悉 ECharts 各个数据图的代码规范。
2. 根据具体应用需求，学会 ECharts 图表样式的选择与使用操作。

**【实训内容】**

将任务实施中的各个操作，均选用不同的 ECharts 图表样式进行数据的可视化操作。

# 任务 6.3 使用 D3.js 可视化数据

## 任务描述

1. 借助学习论坛、网络视频等网络资源和各种图书资源，学习使用 D3.js 可视化数据的操作相关知识。
2. 使用 D3.js，完成岗位技能点的数据可视化操作。

## 任务目标

1. 学会 D3.js 开发环境的搭建操作。
2. 学会使用 D3.js 对岗位技能点图谱做树状图可视化操作。

## 知识准备

数据可视化作为大数据应用的一个分支，随着大数据时代的到来得到了飞速发展。各种各样的开发库如雨后春笋般出现，单是前端方向就不下百种。D3.js 无疑是此中翘楚，是在世界范围内最受广泛认可的前端可视化工具，其在 Github 上的排名居前。

D3.js 受到关注的原因是其具有高自由性、代码简洁和有研究的趣味性 3 点特性。

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



笔记

作为一种完全开源的技术，它不追求作图的速度，有很多为高端开发者考虑的地方，使其成为一项“值得深入学习”的技术。绘图，本身就有很强的趣味性，开发人员得到的结果是图形而不是枯燥的数字，在学习中也不易感到烦闷。

D3（Data-Driven Documents）是一个 JavaScript 库，也叫作 D3.js，它可以通过数据来操作文档。D3 可以通过使用 HTML、SVG 和 CSS 把数据鲜活形象地展现出来。D3 严格遵循 Web 标准，因而可以让开发者的程序轻松兼容现代主流浏览器并避免对特定框架的依赖。同时，它提供了强大的可视化组件，可以让使用者以数据驱动的方式去操作 DOM。

D3.js 需要以下预备知识。

① HTML：超文本标记语言，用于设定网页的内容。

② CSS：层叠样式表，用于设定网页的样式。

③ JavaScript：一种直译式脚本语言，用于设定网页的行为。

④ DOM：文档对象模型，用于修改文档的内容和结构。

⑤ SVG：可缩放矢量图形，用于绘制可视化的图形。

## 任务实施

搭建一个简易的 D3.js 开发环境，步骤如下：

① 准备阶段。在开始前，确保已经安装好一个文本编辑器。

② 搭建环境。可以在 http://d3js.org/网站下载最新版本的 D3.js，也可以在 https://github.com/mbostock/d3/tags 网站下载之前的版本。另外，如果对开发中的最新 D3 版本感兴趣，可以 fork（服务端的仓库克隆）以下的代码库 https://github.com/mbostock/d3。

微课：6-3 使用 D3.js 可视化数据

下载并且解压缩后，将得到 d3.v3.js、d3.v3.min.js 和许可证文件 3 个文件。在开发过程中，建议使用 d3.v3.js，它可以帮助开发者深入到 D3.js 库中跟踪调试 JavaScript 代码。把 d3.v3.js 和新建的 index.html 放在同一个文件夹里，index.html 应该包含下面的代码。

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
<title>Simple D3 Dev Env</title>
<script type="text/javascript" src="jquery.js"></script>
    <script type="text/javascript" src="d3.v3.js"></script>
</head>
<body>
</body>
</html>
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



③ 编程实现。树状图（Tree）通常用于表示层级、上下级、包含与被包含关系。接下来将通过 D3. js 来实现本案例，如图 6-11 所示。

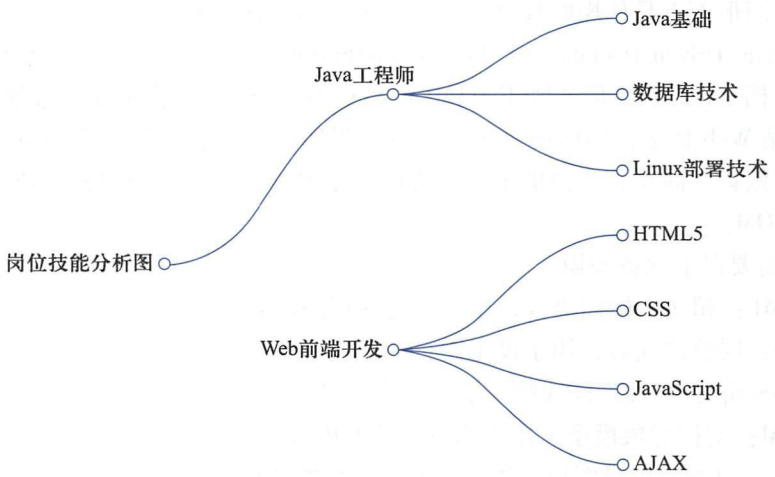



图 6-11 岗位技能点的可视化

代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>d3 tree</title>
    <style type="text/css">
        /* svg style */
.node circle {
    fill:#fff;
    stroke:steelblue;
    stroke-width:3px;
}

.node text {
    font:12px sans-serif;
}

.link {
    fill:none;
    stroke:#ccc;
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
        stroke-width:2px;
    }

        </style>
    </head>
    <body>
    <div id="chart"></div>
    <script type="text/javascript" src="js/jquery-1.8.3.min.js"></script>
    <script type="text/javascript" src="js/d3.min.js"></script>
    <script type="text/javascript" src="js/tree.js"></script>
    </body>
    </html>

    Json:
    {
            "name":"岗位技能分析图",
            "children":
            [
                {
                  "name":"Java 工程师",
                  "children":
                  [
                      {"name":"Java 基础"},
                      {"name":"数据库技术"},
                      {"name":"Linux 部署技术"},
                  ]
                },
                {
                  "name":"Web 前端开发",
                  "children":
                  [
                      {"name":"HTML5"},
                      {"name":"CSS"},
                      {"name":"JavaScript"},
                      {"name":"ajax"}
                  ]
                },
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
            {
              "name":"云计算工程师",
              "children":
              [
                  {"name":"Linux 基础"},
                  {"name":"OpenStack 基础"},
                  {"name":"Python"},
                  {"name":"SQL      "}
              ]
            }
        ]
        }
```

tree.js:

```
var width=400;
var height=400;
var tree=d3.layout.tree()
.size([width,height-200])
.separation(function(a,b){return (a.parent==b.parent?1:2)/a.depth;});
    var diagonal=d3.svg.diagonal()
        .projection(function(d){return [d.y,d.x];});
    var svg=d3.select("#chart").append("svg")
        .attr("width",width)
        .attr("height",height)
        .append("g")
        .attr("transform","translate(100,0)");
    d3.json("",function(error,root){
        root=jsondata;
        var nodes=tree.nodes(root);
        var links=tree.links(nodes);
        console.log(nodes);
        console.log(links);
        var link=svg.selectAll(".link")
          .data(links)
          .enter()
          .append("path")
          .attr("class","link")
          .attr("d",diagonal);
```

非卖品，仅供非商业用途或交流学习使用

版权所有，严禁传播，违者自负法律责任！



```
var node=svg.selectAll(".node")
    .data(nodes)
    .enter()
    .append("g")
    .attr("class","node")
    .attr("transform",function(d){return "translate("+d.y+","+d.x+")";})

node.append("circle")
    .attr("r",4.5);

node.append("text")
    .attr("dx",function(d) { return d.children ? -8:8;})
    .attr("dy",3)
    .style("text-anchor",function(d){return d.children ?"end":"start";})
```

## 项目实训

**【实训任务】**

利用 D3.js 实现数据的可视化操作。

**【实训目的】**

1. 熟悉 D3.js 各个数据图的代码规范。
2. 根据具体应用需求，学会 D3.js 模板的选择与使用操作。

**【实训内容】**

将任务实施中的各个操作，均选用不同的 D3.js 模板进行数据的可视化操作。

## 项目小结

本项目通过使用 3 种数据可视化的方法，对于数据采集、清洗以及分析环节得到的数据进行了直观地展现。Excel 的优势在于不用去自己封装数据格式，直接将数据导入并保存在 Excel 工作簿中，即可通过合适的图标类型完成数据的可视化操作。Ecarts 是由百度公司开发的，其中提供了许多有关中国地图的可视化案例模板，非常适用于与中国地区信息数据的可视化。其使用的数据格式为 json 格式，在生产环境下需要在项目工程中对数据库的数据进行封装后提供给前端使用。D3.js 与 ECharts 类似，也是一个前端开发工具，需要将数据封装成 json 格式传递到前端。相比较而言，D3.js 拥有更为丰富的图库，在世界范围内更受前端开发者的欢迎，因此它会不断提供更多有趣的模板与更新版本。

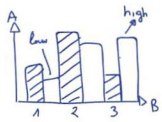
非卖品，仅供非商业用途或交流学习使用

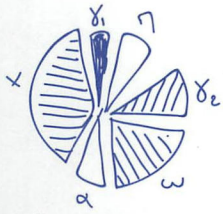
# 参 考 文 献

[1] 谭磊，范磊．Hadoop 应用实战［M］. 北京：清华大学出版社，2017.
[2] 陆嘉恒．Hadoop 实战［M］. 2 版．北京：机械工业出版社，2012.

**CETC 云计算技术与应用专业校企合作系列教材**

- ◇ Android 云存储客户端开发
- ◇ 软件定义网络（SDN）技术与实践
- ◇ Docker 容器技术与应用
- ◇ 云存储技术与应用
- ◆ Hadoop 大数据平台构建与应用
- ◇ 云计算数据中心运维
- ◇ 虚拟化技术与应用
- ◇ 云计算网络技术与应用
- ◇ Python 语言



ISBN 978-7-04-049100-5

定价 24.80 元